

# Modelagem de processos de negócios com BPMN

## Curso completo



Glauco S Reis  
Editora PortalBPM

## Eventos



Final



Intermediário



Início



Término



Mensagem



Link



Cancelamento



Dados



Complexo



Exceção



Compensação



Timer

## Gateways



AND



AND



Complexo



eventos



OR



XOR

# **Modelagem de processos de negócios com BPMN**

## **Curso completo**



Glauco S Reis  
Editora PortalBPM

Copyright @ 2008 by Editora PortalBPM Ltda.

Direitos desta edição reservados a  
Editora PortalBPM Ltda  
Av. Sta Catarina, 1396 – sala 4  
CEP : 04367-050

Impresso no Brasil por HR Gráfica

Proibida a reprodução, total ou parcial, por qualquer meio ou processo, Seja fotográfico, gráfico ou microfilmagem etc.

Estas proibições aplicam-se também nas características gráficas e/ou editoriais.

A violação dos direitos editoriais é punível como crime (Código penal, art 184, Lei 6.895/80), com busca, apreensão e indenizações diversas (Lei 9610/98 – Lei dos direitos autorais – arts. 122, 123, 124 e 126)

Reis, Glauco dos Santos

Modelagem de processos de negócios com BPMN – Curso completo /  
Glauco S. Reis;

Revisão técnica – equipe PortalBPM

São Paulo; Editora PortalBPM Ltda; 2008

1. Modelagem de processos de negócios 2.BPMN 3.BPMS



# Prefácio

Estamos vivenciando um período de mudanças profundas, em áreas de conhecimento distintas como processos para desenvolvimento de sistemas e gestão empresarial.

Toda uma nova gama de conceitos e ferramentas estão sendo apresentadas e definidas, e que provavelmente deverão nortear o futuro da TI nos próximos anos.

Um destes conceitos, e provavelmente o que gera a maior concordância entre os especialistas é a notação BPMN.

A sigla BPMN é o acrônimo de "*Business Process Modeling Notation*", ou notação para modelagem de processos de negócios.

Embora quase não exista discordância quanto à efetividade do padrão, as documentações são limitadas e a especificação oficial é densa e intrincada.

A proposta deste livro é explorar a notação BPMN, como descrita na especificação oficial, sem as particularidades introduzidas por cada fabricante.

Toda a simbologia será explorada, e estaremos fornecendo interpretação para cada símbolo, erros comuns no desenho, além de discutirmos um pouco sobre mapeamento de processos e **design patterns** aplicados à notação.

Espero que você aproveite a leitura, e que muitas das dúvidas possam ser sanadas ao longo do livro.

Glauco Reis  
glauco@portalbpm.com.br

# Sumário

1 - Prefácio	
2 - Ferramentas BPMS	- 06
2.1- Workflows	- 06
2.2 - BPMN	- 07
2.3 - BPEL	- 08
2.4 - SOA e Web Services	- 09
2.5 - Dashboards	- 10
2.6 - Realinhamento dos processos de negócios	- 11
2.7 - Business Process Management Systems	- 11
2.8 - Pure players	- 15
2.9 - BPMN e Futuro	- 15
3 - Processos	- 16
3.1 - UML e BPMN	- 16
4 - Tipos de diagramas BPMN	- 18
5 - Pools e Lanes	- 20
5.1 - Instâncias de processos	- 21
6 - Eventos	- 22
7 - Eventos de timer	- 23
8 - Eventos de mensagem	- 26
9 - Eventos de dados	- 29
10 - Controles de exceção e compensação	- 31
11 - Conexão entre diagramas	- 35
12 - Terminadores do processo	- 35
13 - Resumo dos tipos de eventos da BPMN	- 37
14 - Gateways	- 38
15 - Eventos do tipo XOR (OU exclusivo)	- 38
16 - Token	- 39
17 - Eventos AND (E)	- 41
18 - Eventos OR (OU)	- 41
19 - Tratamento de eventos	- 41
20 - Eventos múltiplos	- 42
21 - Eventos Complexos	- 43
22 - Gateways diretamente em fluxos	- 44
23 - Atividades e tarefas	- 46
24 - Loop	- 46
25 - Múltipla instância	- 48

26 - Compensação	- 49
27 - Ad Hoc	- 50
28 - Pools e Lanes	- 52
29 - Artefatos	- 56
30 - Dados	- 56
31 - Grupos	- 58

## Parte II

### **ERROS COMUNS NO DESENHO BPMN - 59**

01 - Gateways de tipos diferentes na junção e bifurcação-	60
02 - Gateways baseados em eventos	- 62
03 - Utilização de elementos de fluxos com fluxos condicionais-	63
04 - Eventos	- 65
05 - Fluxos, pools e lanes	- 68
06 - Fluxos, pools e lanes	- 69
07 - Levantamento dos processos de negócios	- 70
08 - Processos e Macro-processos	- 71
09 - Os fluxos de processos	- 74
10 - "As is" e "To be"	- 75
11 - Use Cases e processos	- 76
12 - StoryBoards	- 78
13 - BPEL e BPMN	- 79
14 - O Mapeamento BPMN para o BPEL	- 82
15 - Editor BPMN da revista PortalBPM	- 85
16 - Outras ferramentas de mercado	- 93
17 - BPMN design patterns	- 94

## **2 - Ferramentas BPMS**

Antes de iniciarmos nossa jornada pela notação BPMN, vamos apresentar como a TI evoluiu até chegamos ao momento atual, com as chamadas soluções BPMS.

As disciplinas de modelagem de processos tiveram grande foco ao final da década de 80 e início da década de 90, com estudos de cientistas como Michael Porter. Nesta época os estudos mantinham foco em administração de empresas, centrando esforços em como as empresas poderiam ser melhoradas em termos de estrutura organizacional. O que se percebia é que o processo de construção das empresas normalmente se fazia de forma desestruturada, com novos departamentos e processos sendo criados sob demanda. Em um mundo não globalizado, isto até que não afetava muito a saúde da empresa. Mas com a internet e globalização, empresas grandes começavam a competir com empresas pequenas de igual para igual, e a necessidade por uma redução de custos e melhoria de processos junto ao cliente se tornou necessária como forma de garantir sobrevivência em um ambiente competitivo. Nesta época, o foco dos estudos era como melhorar os processos, e embora houvessem vários esforços isolados por parte da TI para solução destes problemas, estes esforços ainda não estavam integrados para formar o conceito que hoje conhecemos como soluções BPMS.

### **2.1 - Workflows**

Eram muito conhecidas na década de 80 como “ferramentas de colaboração”. Uma implementação de destaque foram as soluções da Lotus, que acabaram sendo incorporadas posteriormente aos produtos da gigante IBM.

A necessidade por organizar de alguma forma a comunicação entre as pessoas, transferindo responsabilidades ao longo do processo era e até hoje é fundamental em qualquer tipo de empresa. O nome das ferramentas de colaboração se tornou chamativo, e o conceito de workflow se firmou e se mantém até hoje.



Na verdade, inúmeras empresas vendem soluções de workflow como se fossem soluções BPMS hoje no mercado, mas há diferenças que até hoje os especialistas tentam explicar. De uma forma geral, um workflow é uma das peças que compõe uma solução BPMS.

Em uma ferramenta de workflow existem dois conceitos principais : o motor de execução e a ferramenta de definição de processo ou fluxo.

Estas ferramentas de definição inicialmente foram criadas de forma textual, indicando os participantes e em que ordem eles executariam atividades. Com o tempo, conforme as próprias notações para processos evoluíram, todo um conjunto de notações gráficas, como IDEF, BPMN, UML, grafos, entre outros foram utilizadas como forma de representar visualmente estes processos em execução em um Workflow.

O consorcio WfMC foi notadamente um destaque, propondo uma especificação utilizada até hoje por muitos provedores de solução BPMS.

## **2.2 - BPMN**

Entre os padrões comentados, a notação de destaque na atualidade é o BPMN (Business Process Modeling Notation), que estaremos explorando nos próximos capítulos deste livro. Além de mais moderna que notações como IDEF e UML, também possui um rico set de elementos gráficos para representação de uma série de situações que acontecem nos fluxos dos processos. O padrão está em evolução e longe da perfeição, mas de todas as notações da indústria é o que mantém a maior concordância da indústria.

Infelizmente, o padrão BPMN é uma notação gráfica para desenho de processos, e não está associado a nenhum formato de armazenamento específico.

Houve uma tentativa de utilizar a força da organização que estava com credibilidade muito forte pela criação do padrão no sentido de firmar outro padrão, o BPML (*Business Process Modeling Language*), que seria um formato binário para armazenamento e execução dos processos criados em BPMN. Esta tentativa foi frustrada por outro padrão já em estágio mais avançado de adoção, chamado BPEL.

## 2.3 - BPEL

A linguagem BPEL (*Business Process Execution Language*) é hoje uma forte candidata a ocupar o padrão como notação para execução de processos. Infelizmente, da mesma forma que um motor de fórmula 1 não pode ser colocado sem inúmeras adaptações em um veículo comum de mercado, o mesmo pode-se dizer em relação ao BPMN mapeado para BPEL (com o BPMN como o motor neste exemplo).

A notação BPMN é muito completa e sua evolução se deu em torno das necessidades que aparecem no dia a dia dos desenhos dos processos, enquanto que o BPEL foi criado como um padrão para permitir que um motor pudesse orquestrar fluxos baseados em Serviços Web (Web Services).

Em outras palavras, os caminhos seguiram em paralelo e com focos diferentes em mente. Há todo um esforço no sentido de tornar o BPEL ideal como formato de armazenamento dos desenhos BPMN, mas hoje os fabricantes adotam duas formulas para o mapeamento :

1) Reduzir o número de elementos gráficos disponíveis, para manter a conversão para o BPEL viável. Isto ajuda se tivermos em mente apenas a execução do processo, mas limita o analista de negócios, se a ferramenta de desenho for a mesma ferramenta que irá operacionalizar os fluxos.

2) Incorporar novos elementos ao BPEL, permitindo que toda e qualquer representação BPMN possa ser desenhada sem problemas. Isto traz um inconveniente ao BPEL, pois o mesmo foi criado para ser totalmente interoperável, e no momento em que cada um dos fabricantes incorpora novidades, os motores para execução tendem a se tornar incompatíveis.

O que se espera nos próximos anos é que uma evolução nos dois padrões venha a surgir, ou que outro formato de armazenamento se firme como padrão para desenho. Ainda há uma perspectiva futura que é a criação da notação BPDM, um formato intermediário entre o BPEL e o BPMN.

## **2.4 - SOA e Web Services**

Anteriormente comentamos que o BPEL foi criado como um padrão para orquestração de serviços WEB. Esta foi outra grande inovação introduzida a partir das limitações encontradas nos **paradigmas atuais da orientação para objetos**. Por mais que tentemos reaproveitar os códigos criados a partir da OO, o grau de acoplamento entre eles ainda é muito forte, e torna difícil o reaproveitamento de códigos de forma corporativa.

A arquitetura WEB Java, com seus módulos EAR, WAR, JAR, SAR, RAR, etc. também não facilitam o reaproveitamento de códigos.

A idéia dos Web Services, que posteriormente foi promovida a arquitetura SOA, propõe que os componentes sejam criados e “soltos” dentro de servidores de aplicação, de forma que diversos programas se beneficiem de sua utilização, sem que necessitem ser incorporados fisicamente a um programa específico.

Se esta promessa de componentização se concretizar, teremos um novo patamar de reutilização na indústria. Não há garantias de que este terá sucesso, mas os indicadores atuais são muito promissores.

Bom, imaginando um cenário ideal teremos os desenhos dos fluxos sendo criados em BPMN e salvos ou mapeados para BPEL, que por sua vez fará com que serviços web sejam orquestrados. Então já temos tudo o que é necessário para uma solução BPMS ? Bom, nem tudo ainda.

## 2.5 - DashBoards

O sonho de consumo de qualquer empresário é poder controlar de forma ativa sua empresa, identificando como os processos estão durante a execução, tempos de execução de cada atividade, e outras informações. Se todas estas informações puderem ser apresentadas em gráficos de fácil compreensão, melhor ainda.

É um conceito semelhante ao de BI (Business Intelligence), onde dados são analisados nas empresas e medidas são tomadas para melhoria dos processos.

A idéia de BI ou mineração de dados trabalha com dados após o fato estar consumado, ou seja, após o gasto já ter concretizado. Este controle a que nos referimos é pró-ativo, ou seja, devemos coletar dados ao longo da execução de um fluxo de processo, e estudarmos os dados antes mesmo do término da execução do processo.

Este estudo precisa ser feito de forma facilitada, pois temos pouco tempo para tomada de decisão. Por isso deve ser visual em formato de gráfico.

Nas soluções BPMS, colocamos termômetros dentro do fluxo, e medimos estes termômetros apresentando gráficos que sumarizam a evolução do processo.

Isto é chamado de Dashboard, e é uma peça fundamental das soluções BPMS. Precisamos corrigir eventuais distorções no fluxo bem antes do final, para que possamos garantir a economia e redução nas perdas. Sem este termômetro, é praticamente inviável descobriremos se há algo errado ou mesmo onde devemos corrigir no processo durante sua execução. Será necessário a execução do processo para que possamos depois da execução encontrar as falhas.

Esta melhora precisa acontecer a todo momento, e um termo muito utilizado para descrever esta melhoria contínua é “realinhamento dos processos de negócios”.

## 2.6 - Realinhamento dos processos de negócios

Com todas estas ferramentas dentro da empresa, com os desenhos criados com BPMN, sendo orquestrados por um workflow, executando Web Services com BPEL, e que vão sendo analisados momento a momento para encontrarmos possíveis problemas e pontos de melhoria, através dos Dashboards, precisamos de um mecanismo para podermos realinhar rapidamente, ou seja, resolver rapidamente estes problemas para que o fluxo continue fluindo da forma esperada.

Para isto, não é tolerável aguardar meses por alterações na TI, para que um código funcione como esperado.

O que se espera das melhores soluções BPMS é que seja muito fácil introduzir modificações, com poucas alterações e facilidade nas alterações dos códigos. Isto não é um conceito simples de implementar, e parece que quanto mais visuais as ferramentas se tornarem, mais rapidamente conseguiremos este realinhamento desejado.

## 2.7 - BPMS

### Business Process Management Systems

Pois bem, uma solução BPMS reúne as inovações e tecnologias anteriormente citadas. Uma definição para o BPMS poderia ser :

**“Uma solução BPMS permite a geração e controle dos processos de negócios da empresa, proporcionando rápida tomada de decisão e realinhamento dos processos de negócios de forma agilizada.”**

Um pouco subjetivo não é? Mas é o que o homem de negócio deseja. Em um mundo globalizado e competitivo, os prazos definidos pelas metodologias OO e outras não estão mais se mostrando satisfatórios. Mas podemos ler nas entrelinhas da definição, e deduzir algumas coisas :

- 1) Uma ferramenta BPMS deve permitir o desenho dos processos de forma visual, sendo que as melhores ferramentas suportam o BPMN. As boas ferramentas têm programas próprios de desenho incorporado. Ser visual é importante pois traz agilidade ao gerenciamento do processo. Imagine a quantidade de erros que apareceriam se manipulássemos dezenas ou centenas de arquivos XML de forma manual.
- 2) O suporte ao padrão BPEL4WS não é obrigatório, mas as ferramentas que o suportam fornecem algumas vantagens. Em primeiro lugar, você pode utilizar qualquer editor que gere o formato. Segundo, os processos ficam armazenados em formato XML, o que é mais adequado e proporciona maior interoperabilidade. As desvantagens estão do fato de o BPEL ser orientado para Web Services. Ou seja, não se pode orquestrar uma chamada a um EJB através desta tecnologia, a menos que o EJB seja publicado como serviço.
- 3) É um consenso que as ferramentas devem suportar servidores de aplicações. Algumas criaram os próprios servidores para execução. O ideal seriam ferramentas que suportassem servidores .NET ou J2EE, pois são os mais consagrados. Naturalmente, a tecnologia J2EE permite multiplataforma, e isto por si só já é vantagem. Criar um servidor específico e não utilizar a base J2EE é um erro, pois o grau de maturidade dos servidores J2EE é muito alto. Além disto os servidores incorporam uma série de recursos, como escalabilidade e segurança, e estes recursos podem ser reaproveitados pelos BPMS.
- 4) Toda ferramenta deve permitir o controle do processo. Este controle pode acontecer antes de o processo ser executado (através de simulações) e também durante a execução, para detecção de falhas no processo em operação (Dashboards). Ou seja, antes de publicar o processo devemos poder simulá-lo, para verificar se está como previsto, e durante a execução devemos o ser capazes de extrair métricas para correção com facilidade.

Deve ser possível não só publicar novas versões do processo, como também permitir que as versões antigas continuem enquanto houverem atividades ainda sendo processadas. Isto é necessário, caso contrário será difícil realinhar os processos de negócios rapidamente. Não podemos esperar o término da execução dos processos (o que pode levar dias ou meses), para colocar uma nova alteração em execução. Processos antigos devem continuar executando como previstos, e novos processos devem já ser executados de acordo com o novo desenho.

5) Um recurso muito apreciado pelas áreas gerenciais são os Dashboards, que são gráficos gerenciais, tomados dos processos em execução, e que permitem verificar visualmente quando algo não vai bem. Isto ajuda na tomada de decisão. Uma característica dos Dashboards é que eles tendem a ser muito dinâmicos, e portanto deve ser fácil criá-los e alterá-los sem demandar grandes esforços da TI.

6) Quando dizemos rápido realinhamento nos processos de negócios queremos dizer que as alterações devem se efetivar com custo mínimo de tempo. Uma das vantagens divulgadas pelas ferramentas BPMS é poder gerar resultados mais rápidos do que os conseguidos até então com as metodologias tradicionais. Se para alterar um processo de negócio for necessário acionar uma equipe de desenvolvimento gigantesca, que terão que criar códigos novos para cada Web Service inserido, não estaremos resolvendo um dos problemas que a tecnologia se propõe a resolver, que é gerar uma rápida resposta aos problemas do negócio.

Neste sentido, atualmente existem dois paradigmas sendo adotados pelos grandes fabricantes.

O primeiro é o de ferramentas onde apenas o processo é gerenciado visualmente.

Os componentes de negócios, sejam eles Web Services, CORBA, EJB são criados manualmente, através de ferramentas como Eclipse, NetBeans, WSAD, Sun Studio ou outros, que podem até estar integrados à ferramenta geradora de fluxos.

O segundo tipo de ferramenta, embora não tenha toda a flexibilidade de manipulação de códigos da primeira alternativa, adota um paradigma totalmente visual de construção, onde até mesmo os objetos de negócios são construídos visualmente.

A vantagem deste segundo tipo é a velocidade no realinhamento, com poucos códigos gerenciáveis.

É bem verdade que este tipo de paradigma deve evoluir para algo mais flexível do que os mecanismos atuais. E ferramentas geradoras de código devem evoluir para alternativas onde o processo de construção seja mais ágil.

7) Ferramentas que permitem criar telas visualmente trazem agilidade ao processo de mudança, principalmente quando a atividade exige interação humana.

Um workflow não pode mais ter telas simples, com campos textuais sem formatação. A identidade visual da empresa precisa ser mantida durante a execução do processo. Ou seja, as ferramentas devem permitir a criação de telas complexas, de preferência de forma visual.

8) Ferramentas que suportam apenas componentes Web Services são limitadas atualmente, embora esta seja uma promessa futura. O ideal são ferramentas que agreguem algum tipo de EAI, que permita integrar vários tipos de componentes do legado. Perceba que isto se contrapõe à adoção de BPEL4WS, já que o padrão apenas suporta WS.

9) Um elemento importante, que se esquece com frequência, é a estrutura organizacional da empresa. Antes mesmo de iniciar o entendimento dos processos, se constroem os organogramas da empresa. Uma ferramenta que permita definir organogramas de forma visual, integrando estes organogramas com mecanismos computacionais trazem vantagens.



Infelizmente, a maioria das ferramentas ainda têm limites neste quesito. Manter as informações de usuários disponíveis livremente, em qualquer ponto da corporação, também é uma vantagem.

Quase todas as soluções BPMS mantêm os usuários e senhas em estruturas dentro da própria ferramenta, ao invés de buscar em uma base relacional externa ou servidor LDAP.

Também a maioria não permite edição visual do organograma da empresa.

## **2.8 - Pure players**

Alguns institutos de pesquisa (Gartner Group, por exemplo) se utilizam do termo "*pure player*", para ferramentas que são BPMS puros, sendo que os demais são ferramentas que vêm de algum nicho de mercado pré existente. Algumas ferramentas são Workflows ou EAI's que estão em evolução para o paradigma BPMS. Algumas empresas têm ferramental de desenho (BPMN) e estão evoluindo para se enquadrarem na categoria BPMS. Este termo foi criado para tentar reduzir a panacéia em relação aos diversos fornecedores que desejam ter sua ferramenta seja considerada BPMS "*pure player*". Uma ferramenta do tipo "*pure player*" tende a ser melhor, já que foi criada com o espírito destas idéias desde o início.

## **2.9 - Futuro e BPMN**

Vale a pena gastar esforços no padrão BPMN agora? Bem, se você for um analista de negócios este provavelmente será o padrão dominante para desenho nos próximos anos. Se você é da área de TI, é muito provável que a operacionalização dos fluxos seja feita através de uma evolução do BPMN, com mais elementos programáticos incorporados. Mesmo que o padrão dominante continue sendo a UML, há uma perspectiva de evolução da notação para que incorpore o BPMN como uma evolução dos diagramas de atividade.

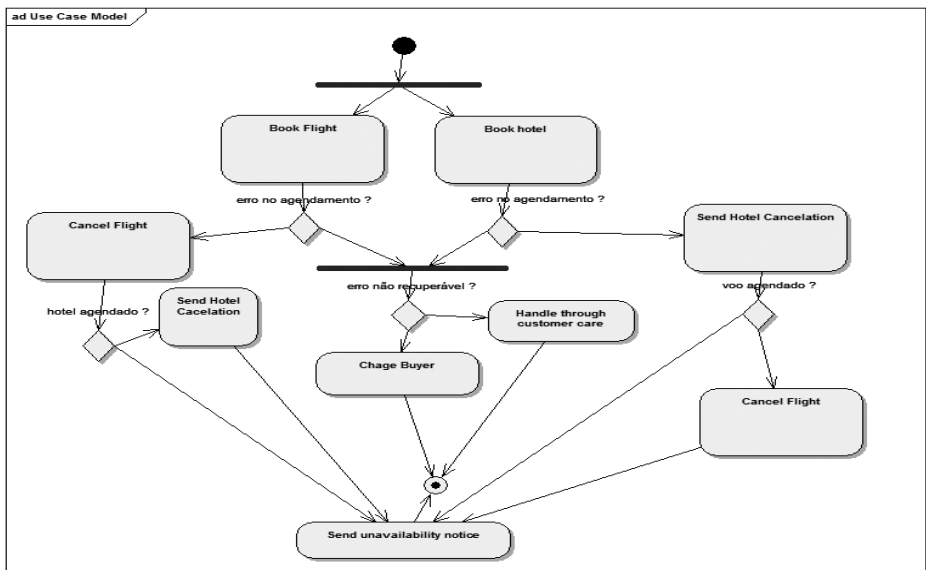
## 3 - Processos

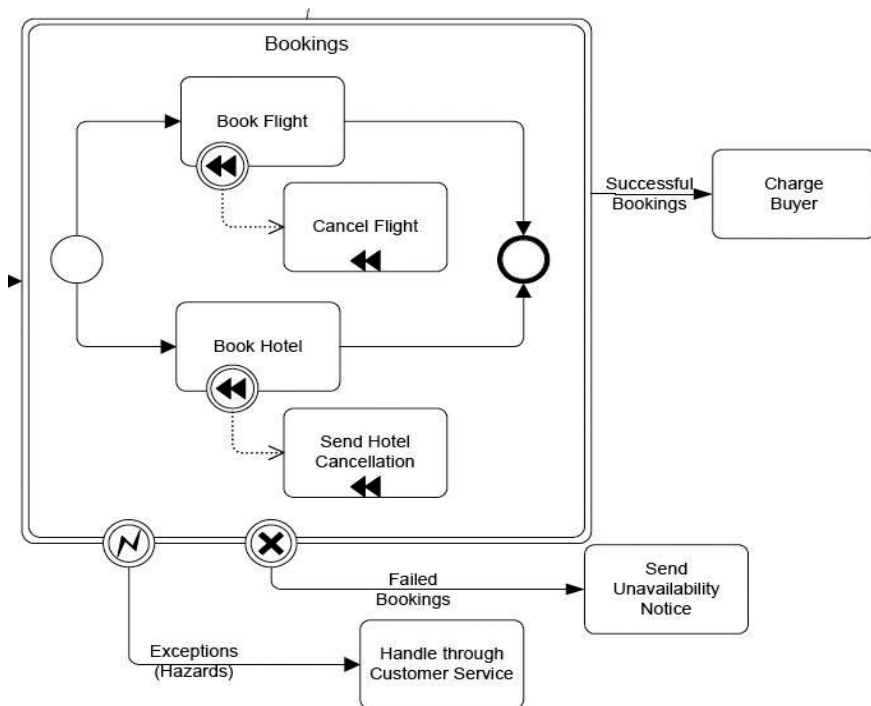
### 3.1 - UML e BPMN

Para que necessitamos de mais uma notação para desenho de processos? Será que as notações existentes, como os diagramas de atividade da UML, já não contemplam também este tipo de representação ?

Eles representam muito bem etapas na execução de um "Use Case", ou mesmo um trecho mais complicado de uma rotina. Os fluxos de processos de negócios têm se mostrado muito mais complexos a cada dia, e a notação UML embora seja muito formal, deixa a desejar em termos de diversidade gráfica, além de utilizar as mesmas simbologias para o tratamento do negócio e para tratamentos de erros, que normalmente não fazem parte do negócio em si. Em diagramas maiores a tendência é confundir e dificultar a visualização e manutenção dos fluxos.

Para tornar isto mais claro (e visual), vamos utilizar o exemplo de apontamento de viagem, explorado na documentação oficial BPMN. Veja o mesmo exemplo modelado de duas formas :





BPMN

O que se pode observar em primeiro lugar é que, no caso da UML, como não existem mecanismos de compensação e geração de erros, somos obrigados a utilizar elementos de decisão para indicar o fluxo em caso de erro.

Neste caso fica mais difícil identificar qual elemento está sendo utilizado como parte do tratamento do negócio, e qual decisão foi inserida como desvio para tratamento de erros, dificultando a manutenção.

O mesmo irá acontecer quando outros elementos, como timers e mensagens são adicionados ao diagrama. Eles isolam elementos de interação dos processos de tratamentos da lógica de negócios, de forma mais clara. Além disto, o próprio fato de a imagem já ser representativa, torna mais claro o entendimento pelo cérebro do que aquele desvio ou evento faz.

Em resumo, um diagrama BPMN torna mais claro o entendimento e adiciona elementos gráficos que facilitam a compreensão do que está acontecendo no diagrama.

## 4 - Tipos de diagramas BPMN

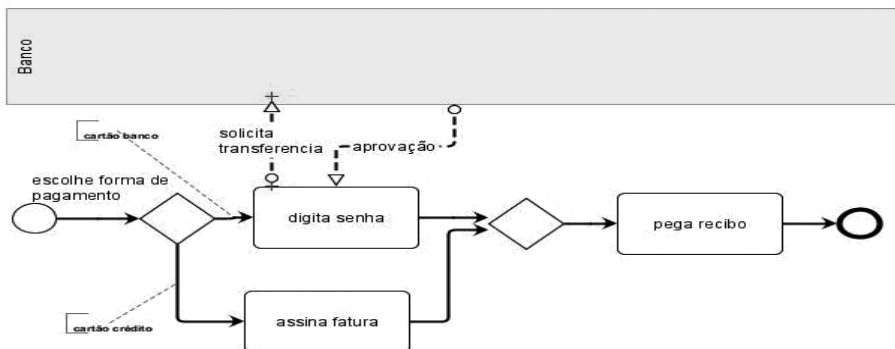
Cada um dos tipos de diagrama é chamado de BPD ("***Business Process Diagram***"), mas têm nomes mais específicos, de acordo com seu detalhamento.

Algumas vezes desejamos entender em profundidade como um fluxo opera, e outras vezes desejamos apenas transmitir ao leitor do diagrama como dois fluxos distintos operam. Para permitir esta divisão em categorias temos três tipos de diagramas :

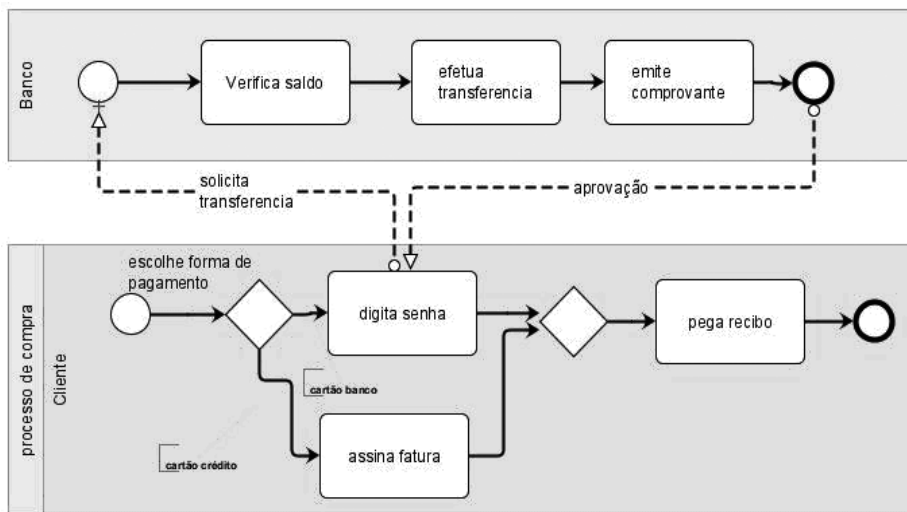
Os "***Private Business Process***" ou Diagramas de processos privados, que utilizamos quando não é relevante representarmos como diferentes fluxos interagem. Estamos preocupados apenas com o teor deste fluxo em si.



Existem os "***Abstract process***" ou Processos abstratos. Em um processo abstrato, não estamos preocupados com o conteúdo do fluxo em si, mas sim como ele colabora com outros fluxos dentro de um sistema.



Já nos “*Colaboration Process*” ou Processos colaborativos desejamos obter um grau maior de detalhamento, apresentando como dois ou mais fluxos se comunicam.



Podemos imaginar estes três tipos de diagramas como sendo um só, onde características podem ser apresentadas ou não, dependendo do que desejamos visualizar. Imagine um programa de CAD com uma imagem de um veículo, onde podemos visualizar sua aparência externa (equivalente ao diagrama abstract) ou as peças que o compõe (equivalente ao diagrama private). A especificação não dita regras de como cada implementação de ferramenta deve operar, mas parece que a forma mais intuitiva seria a de que cada diagrama pudesse ser chaveado para apresentar seus elementos internos ou não. Um aspecto ainda a explorar é como estes tipos de diagramas afetam o mapeamento para um fluxo de processos.

Um processo BPEL é representado por um processo privado, por exemplo. Para que tenhamos a colaboração entre processos em BPEL, necessitamos de mais de um arquivo BPEL, cada um mapeado para um processo privado. Em resumo, um diagrama BPMN pode ser mapeado para mais de um arquivo BPEL. Normalmente um arquivo BPEL é mapeado para apenas um diagrama BPMN.

## 5- Pools e Lanes

Conforme um fluxo vai sendo executado, ele passa por várias atividades.

Em um mundo real e colaborativo, as atividades são atribuídas a pessoas ou perfis diferentes, ao longo de sua execução. Documentos necessitam ser aprovados por gerentes ou coordenadores, e vendedores necessitam inserir pedidos no sistema.

Cada um destes “perfis” são identificados dentro do BPMN em elementos chamados pools ou lanes. A pool seria a piscina onde os competidores nadam. As lanes são as raia, onde cada nadador irá se apresentar. O resultado do processamento, neste caso, será o conjunto de processamentos das várias lanes (o nadador que chegar em primeiro lugar do outro lado).

O conceito de pools e lanes está muito preso aos mecanismos de segurança das soluções BPMS. Um vendedor pode inserir um pedido, mas apenas um coordenador pode aprovar uma compra através de cartão de crédito. Quando o fluxo é modelado já se considerando os pools e lanes, fica mais fácil no futuro organizarmos os vários perfis que irão executar cada uma das atividades.

Mas, qual a diferença entre um pool e uma lane ?

Bem, o pool define um fluxo de processo, onde em seu interior o processamento ocorre independentemente do que acontece ao seu redor. Um pool é como se fosse um módulo ou programa independente de um sistema. Imaginando o exemplo de uma piscina, se ela estiver dentro de um estádio, várias outras atividades esportivas podem estar acontecendo simultaneamente, mas a competição dentro da piscina independe destas outras atividades que acontecem simultaneamente no estádio. Já uma lane define o perfil que está executando a atividade em dado momento. Via de regra, uma pool define um processo, enquanto que a lane define os participantes que irão concretizar este processo.

## **5.1 - Instâncias de processos**

Imaginando que temos um desenho de processo pronto, surge a dúvida do que seria uma instância de processo. Uma instância é uma execução isolada de um processo.

Um programa instalado em um computador é como se fosse um processo. Já as várias instâncias em execução seriam os programas executando em dado momento em um computador.

Cada vendedor dentro de uma loja, atuando em seu processo de vendas é uma instância de processo isolada. Todos seguem os mesmos passos, mas cada um de forma independente.

A longevidade de uma instância de processo é muito variada, podendo ir de minutos a décadas para processamento.

Ou seja, uma vez iniciado um processo, ele se mantém (ou deveria se manter) em execução até que seu término fosse atingido.

Se estivermos imaginando um processo executando em um computador, é importante que esta característica seja preservada, ou seja, se o computador for desligado, quando for reiniciado deve retornar ao mesmo ponto em que estava anteriormente.

De forma similar, diversas instâncias de um mesmo processo podem estar ativas ao mesmo tempo. Uma loja de varejo, onde existam vários vendedores, cada um deles pode estar conduzindo uma instância de processo de vendas, de forma totalmente independente.

Algo deve “acontecer” para que o processo inicie uma instância, e o mesmo para que ele seja finalizado.

Processos são iniciados e terminados através de eventos. Uma vez iniciado, deve acontecer um evento que o finalize.

## **6 - Eventos**

De acordo com a especificação BPMN, os eventos aparecem durante a execução de um fluxo ou são gerados durante sua execução. Somente os eventos têm a capacidade de iniciar ou terminar um processo, mas os eventos não executam tarefas no processo. Eles podem forçar a execução ou mesmo desviar para uma determinada atividade.

A representação gráfica dos eventos é feita por círculos, que podem ser de três tipos:

- 1) iniciais**
- 2) intermediários**
- 3) finais**

Os eventos de início forçam a criação de uma nova instância de execução para o fluxo. Antes dele, não existia esta instância de processo em execução. Algum fato gerador externo irá causar a geração do evento. O início do processo por si só é um evento.

Os eventos intermediários acontecem ou são gerados ao longo da realização de um fluxo já em execução. Podem ser gerados pelo fluxo em execução ou podem ser receptores de um evento gerado por outra instância.

Quando são criados dentro do próprio fluxo (geradores) provavelmente irão gerar alguma notificação a alguma outra instância de fluxo em execução. Quando são gerados externamente ao fluxo de processo em execução (receptores), irão influenciar de alguma forma a execução de outro fluxo.

Existem os eventos finais, que terminam a instância do processo. Após um evento final, nenhuma outra atividade pode ser executada, embora o próprio evento possa gerar informações que afetem outros fluxos em execução.



Devemos ter em mente que os fluxos funcionam como organismos vivos, podendo existir vários deles em execução simultânea, e cada um se comunicando com outros por meio de eventos. A única forma de uma instância de fluxo comunicar-se com outra é mediante o envio e recepção de eventos.

Para podermos diferenciar os eventos de início, intermediário e final, de forma visual, existem representações gráficas distintas para cada um deles. Um evento de início é um círculo com uma linha simples em sua borda, um evento intermediário é representado por um círculo delineado por linhas duplas, e um evento final é delineado por um círculo com linha de espessura dupla. Veja os tipos de eventos existentes.

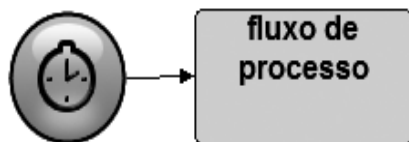


O set completo BPMN define, além desses três tipos, o fato gerador do evento, ou seja, o causador da execução do evento. Estudaremos os vários fatos geradores, iniciando-se com os eventos de timer.

## **7 - Eventos de timer**

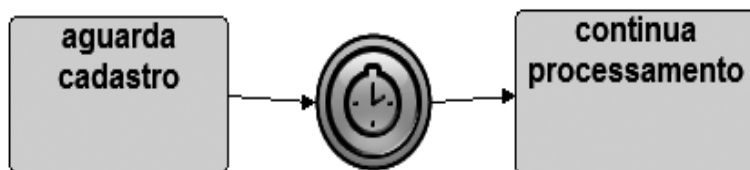
Os timers permitem controlar o tempo ou definir datas para execução de atividades.

Um evento de início de timer indica o início da execução de um fluxo em períodos predeterminados de tempo. Um início de timer pode demarcar uma data específica, como todo último dia do mês (para processamento de uma folha de pagamento); pode indicar uma hora específica, como às dez horas (para iniciar um backup, por exemplo); pode indicar uma data e hora, como dia 23 de janeiro de 2008 (lançamento do livro “Modelagem de Processos de negócios com BPMN”); como também pode ser um evento relativo à data atual (cinco horas contando a partir deste momento).



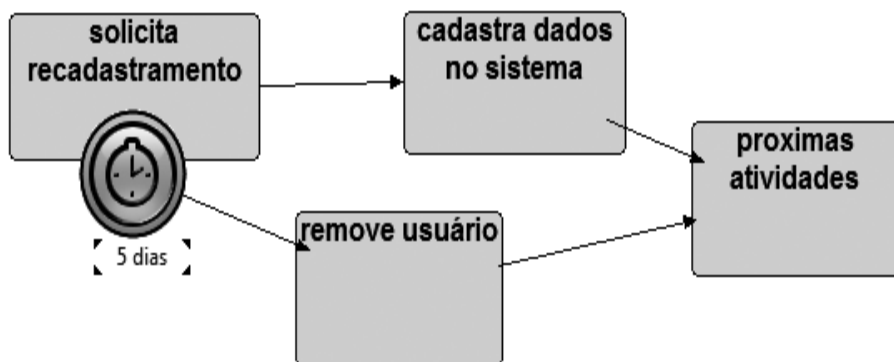
A Figura anterior indica um evento de início. Uma solução BPMS de mercado fica validando, normalmente, cada evento de início, verificando se ele coincide com o momento atual. Quando se encontra uma situação de início, inicia-se uma instância do processo.

Um evento intermediário de timer pode operar de duas formas. Se ele estiver no meio da execução do fluxo, interligando duas atividades, como na figura a seguir, significa um atraso inserido entre a execução das duas atividades.



Isso indica que após a atividade **"Aguarda cadastro"** o fluxo ficará "congelado" pelo tempo definido pelo relógio antes de continuar a próxima execução, que é **"Continua processamento"**.

Mas um timer intermediário pode igualmente estar ligado à borda de uma atividade e irá indicar, nesse caso, um "código de proteção". Se acontecer a condição antes do término da execução da atividade, o processamento será desviado para a atividade que estiver conectada ao fluxo.

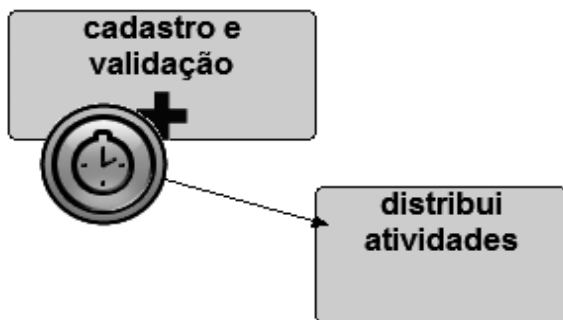


No exemplo anterior, uma atividade solicita o recadastramento, mas pode levar um tempo até que seja executada pelo usuário, assim como pode ficar para sempre no aguardo, já que o usuário pode nunca se cadastrar. Nesse caso, o evento de timer significa dizer que, se o usuário não realizar a atividade em até cinco dias, a atividade **"Remove usuário"** será executada. Caso ela seja realizada dentro desse prazo, **"Remove usuário"** nunca será executada e o processamento continua por **"Cadastra dados no sistema"**. Em qualquer dos dois casos, o processamento continua por **"Próximas atividades"**.

Outra modalidade, nessa representação, seria a de não ter a continuação do fluxo, e forçar sempre a saída pelo mecanismo de exceção.

Nesse caso, mesmo após a execução de **"Solicita recadastramento"**, o fluxo fica no aguardo do final dos cinco dias para continuar por **"Remove usuário"**. Funciona como se o evento estivesse conectado entre as duas atividades.

Basicamente, executa-se um evento timer "atachado" à borda de uma atividade quando se atingiu o período limite ou a data configurada antes do final da execução da atividade. Perceba que pode ser também um sub-processo, com diversas atividades em seu interior, como na figura a seguir.



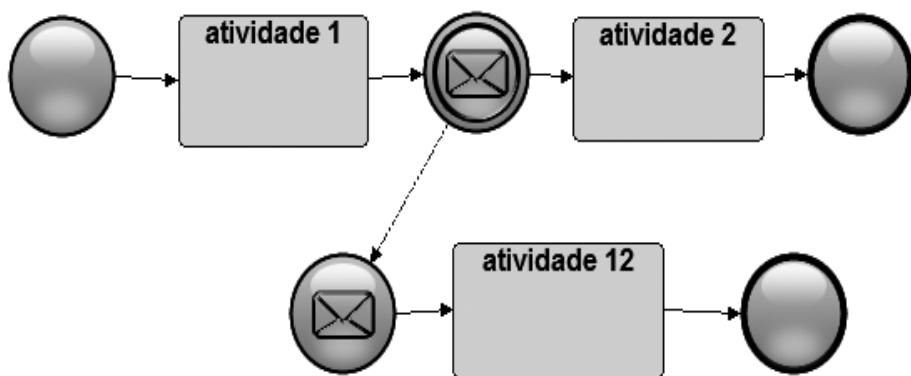
No exemplo da figura anterior, pouco importa o número de atividades existentes dentro de **"Cadastro e validação"**. Quando se atingir o tempo final de cadastramento, cessa o sub-processo **"Cadastro e validação"**, e continua o processamento, por meio de **"Distribui as atividades"**. Forçamos a execução da exceção sempre, pois é a única saída para o fluxo neste caso.

Deve-se tomar cuidado com este tipo de abordagem, pois se o desvio acontecer pelo timer, o sub-processo pode ficar "truncado", e a especificação não dita regras para o que acontece com os códigos já processados dentro da atividade. Não está definido nem faz sentido um evento de timer como finalizador.

## 8 - Eventos de mensagem

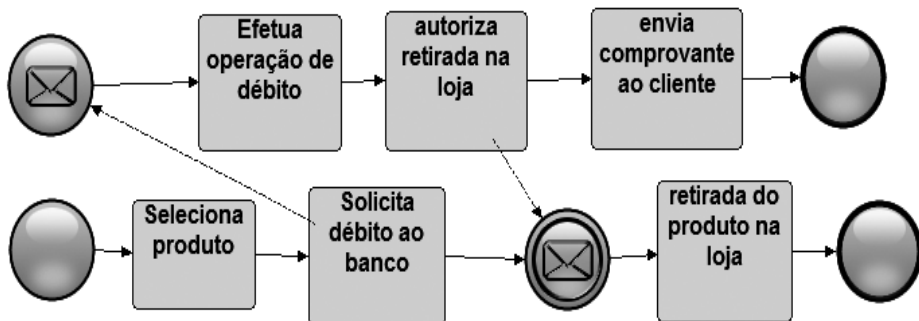
Uma mensagem é um mecanismo de envio de informações entre instâncias de processos. A notação BPMN não define o que se utilizará como tecnologia para envio, como SMTP, JMS ou qualquer outro mecanismo, como também não define o formato das informações a serem enviadas. Uma implementação específica de solução BPMS fará o tratamento mais adequado ao processamento de mensagens.

Os eventos de início de mensagem criam uma nova instância de processo, baseado no recebimento de uma mensagem.



Na Figura anterior, temos dois fluxos de processos. Quando o fluxo superior se inicia e quando a Atividade1 é executada, para um evento intermediário que significa envio de mensagem. Um evento de mensagem entre duas atividades pode ser fonte geradora, como no caso anterior, onde será enviada a mensagem e o fluxo irá continuar, ou pode ser fonte receptora, se uma seta chegar até a mensagem. Neste caso, quando o evento inicial de mensagem for gerado, ele irá forçar a criação do fluxo inferior.

Neste caso, se inicia uma nova instância. Mas após o envio da mensagem no fluxo superior, o mesmo continua a execução da atividade2, e os dois fluxos executam em paralelo e de forma completamente independente. O único momento em que houve comunicação foi no momento do envio de mensagens, que causou a criação de uma nova instância de fluxo.

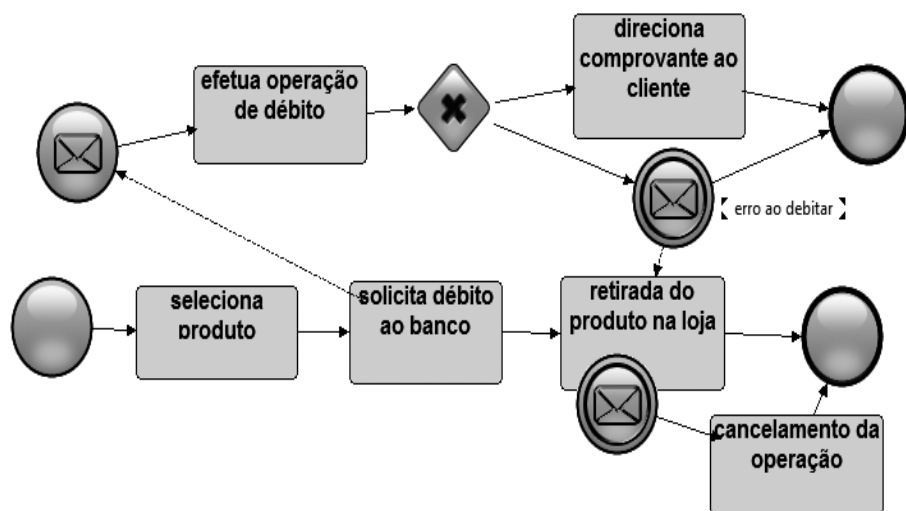


No exemplo da figura anterior, acontece uma série de eventos-mensagem. Primeiro, a partir de uma atividade envia-se uma mensagem que inicia outra instância de processo. Isso é feito em **"Solicita débito ao banco"**. A notação dita que um evento de mensagem na borda deve ser receptor. Uma mensagem saindo diretamente de uma atividade indica que existe um evento de mensagem sendo gerado por esta atividade. Na seqüência do fluxo, a atividade **"autoriza retirada na loja"** envia uma mensagem para o fluxo abaixo, que deveria estar no aguardo para continuar o processamento. Uma vez recebida a mensagem, a atividade **"retirada do produto na loja"** é executada, e os fluxos finalizam.

Embora não esteja claro, os fluxos anteriores são independentes, ou seja, estão posicionados em pools diferentes. Isto pode ser afirmado porque a comunicação entre eles está sendo feita por mensagens, e não seria possível enviar mensagens se as atividades estivessem dentro de um mesmo pool.

Pode-se dizer que são dois fluxos colaborativos.

Quando a atividade está posicionada na borda, como no exemplo a seguir, existe uma outra interpretação possível.



No exemplo anterior, o evento de mensagem na borda da atividade **"Retirada de produto na loja"** está posicionado na borda. Esta representação indica um fluxo de exceção de mensagem. Após a **"operação de débito"**, existe um OR, o qual indica que se tomará um dos dois caminhos. Caso se tome o caminho de erro no processamento (aquele que gera a mensagem), quando a mensagem chega até **"Retirada do produto na loja"**, gera-se um desvio, com base no recebimento dessa mensagem. Nesse caso, a operação é cancelada, visto que o fluxo de exceção foi acionado.

Em resumo, existem dois comportamentos para atividades de mensagem conectadas à borda. Quando o fluxo sai do evento, indica exceção e somente pode ser utilizado posicionada na borda de uma atividade. Quando a mensagem chega até uma atividade, indica que esta atividade estava no aguardo do recebimento para continuar o processamento.

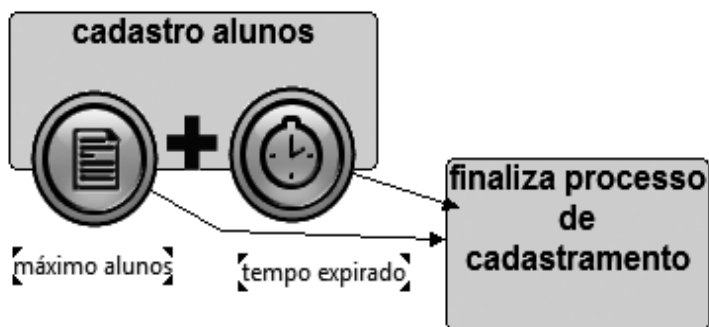
Cabe salientar que esse é o comportamento padrão definido pela especificação, que até deixa alguma dupla interpretação para este tópico.

## 9 - Eventos de dados

Podem-se utilizar eventos de dados como início e intermediário.

Quando utilizados no início de um processo, indicam que se criará uma instância do processo, quando um valor atingir determinado patamar. O modo como os dados são conectados à solução BPMS e o número de vezes que essa solução verifica pelos valores ficam sob completa responsabilidade da implementação BPMS. Lembre-se de que o BPMN é uma notação gráfica, sem pretensões de como será executado.

Um processo automatizado de compra de materiais poderia iniciar um subprocesso de compra de papel para impressora quando fosse detectado, no estoque, a existência de um número baixo de pacotes de folhas. Quando colocado na borda de uma atividade, sempre indica um fluxo de exceção. Ou seja, quando estiver dentro da atividade, se o patamar for atingido, o fluxo seguirá pelo caminho de exceção. Veja um exemplo na figura a seguir.

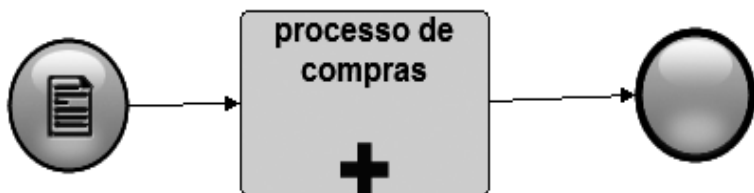


Nesse caso, o sub-processo **"Cadastro de alunos"** fica em execução coletando novos alunos para um novo curso. Ele somente sai desse loop quando uma de duas coisas acontecer.

Ou o tempo para inscrição se encerrou, o que se denota com um timer, ou o número máximo de alunos por sala foi atingido, o que se indica pelo evento de dados. Como não existe continuação do fluxo depois de **"Cadastramento de alunos"**, será necessário que uma das duas exceções aconteça, para que o fluxo continue por meio de **"Finaliza processo de cadastramento"**.

Este tipo de evento pode também ser utilizado como início para um processo ou sub-processo. Por exemplo, quando o limite do estoque está baixo, automaticamente o processo de compras se inicia, como no exemplo a seguir.





Não é permitido que um finalizador gere um evento de dados, ao menos na especificação atual do BPMN.

## **10 - Controles de exceção e compensação**

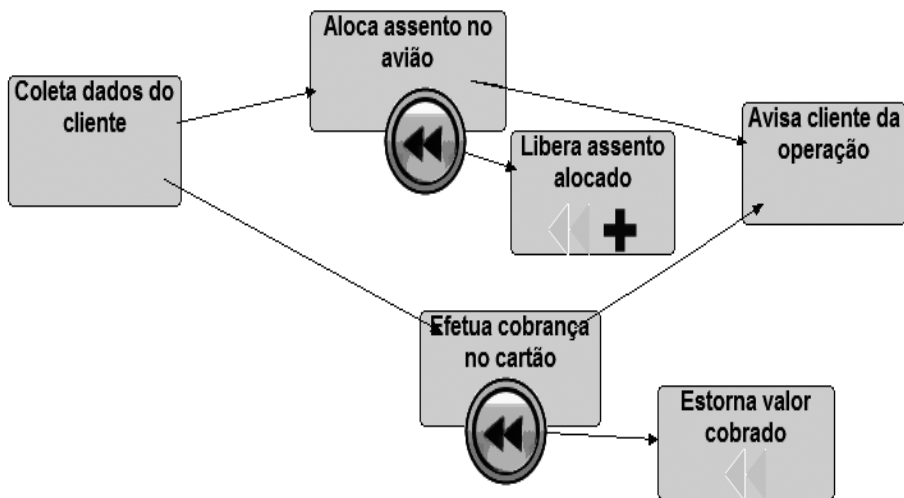
Na maioria das situações, o BPMN será utilizado para orquestrar Web Services. Nesse cenário, surge um problema, pois os Web Services são atômicos e síncronos. Isso quer dizer que, uma vez chamado, um serviço será executado se estiver disponível, e o resultado do processamento persistirá imediatamente. Em resumo, não há rollback de Web Services.

Imagine uma seqüência de chamadas a serviços que precisam operar em forma de bloco, de forma a efetivar todos ao mesmo tempo ou a cancelar todos como uma unidade. Como um serviço é atômico, foi necessário criar mecanismos que permitissem que grupos de serviços pudessem ser “desfeitos”, caso algum deles apresentasse algum problema.

Como parte da especificação BPMN, existe um controle de exceções e compensações criado sob a forma de eventos gerados. Basta conectarmos a borda de uma atividade composta por várias atividades que devem ser atômicas, a um evento especial que atuará como captura de erros que acontecem. Quando se efetuar o desvio para essa outra atividade, os processamentos efetuados poderão ser desfeitos, de modo que se retorne o processamento ao estado inicial.

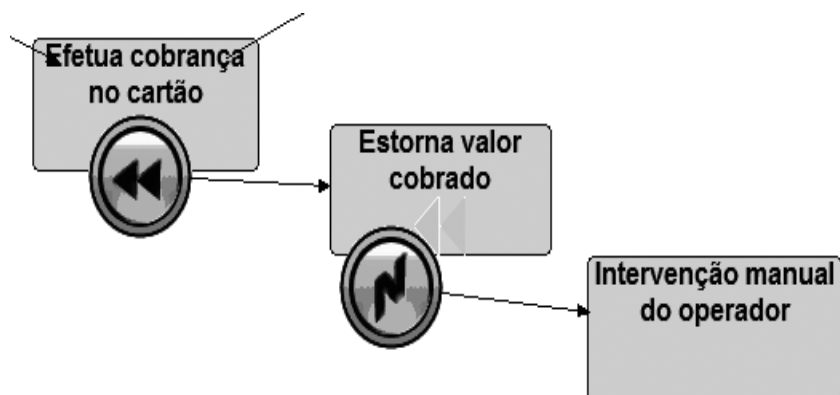
Vale salientar que, nas implementações de bancos de dados atuais, esse mecanismo não funciona como nos rollbacks BPMN, que nem sequer chegam a alterar as informações. Em resumo, as tecnologias de EAI estão mais modernas do que o mecanismo de compensação proposto pela especificação BPMN.

Em vez disso, na implementação de compensação do BPMN os dados são alterados e depois retornam aos valores originais, através de uma atividade especial de compensação, o que pode acarretar efeitos colaterais quando utilizamos serviços a partir de código legado. A figura a seguir ilustra a representação desse mecanismo.

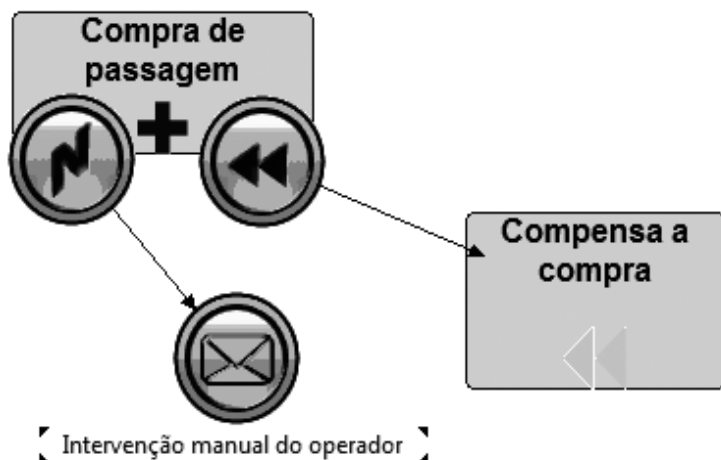


Nesse caso, coletam-se os dados do cliente e efetua-se a compra de passagem. Duas atividades precisam ser executadas: efetuar a reserva no voo e a cobrança do valor. Caso falhe algum desses serviços, acontecerá um desvio para a compensação, que irá garantir que o valor nunca será cobrado ou que o avião não decolará com o assento reservado, porém vazio. Se por outro lado, os dois serviços forem executados com sucesso, o processamento continuará em **"Avisa cliente da operação"**.

Eis uma pergunta de cunho maldoso: e se ocorrer uma falha ao estornar o valor ao cliente? Podemos capturar esse erro e direcioná-lo a uma atividade que irá efetuar o tratamento. Nesse caso, o erro é mais grave e o chamamos de exceção, que precisa de atenção especial. Pode-se representar uma parte do desenho acima da seguinte maneira:



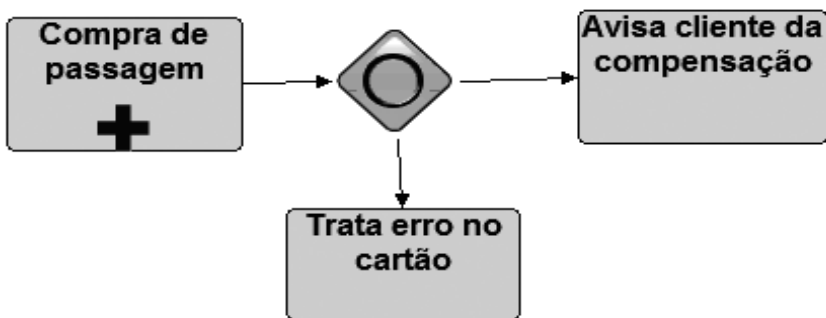
Nesse caso, houve um erro na compensação, e solicitou-se intervenção manual da operadora, a fim de evitar resultados mais graves. Podemos unir a compensação e a exceção em apenas uma atividade, como ilustra a figura a seguir.



Nesse caso, a sub-atividade **"compra de passagem"** pode-se formar por **"pela cobrança da passagem"** e pela alocação. Caso ocorra algum erro no processamento de um serviço, este se desviará para a compensação, que por sua vez tentará desfazer o processo. Se ainda dentro de **"compra de passagem"** ocorrer um erro grave, será gerada a exceção que enviará uma mensagem ao operador do sistema, o qual fará uma intervenção manual.

As compensações são erros não tão graves, onde se tenta corrigir o problema, retornado a um estado possível de execução. Já os erros são mais graves, e exigem um tratamento especial.

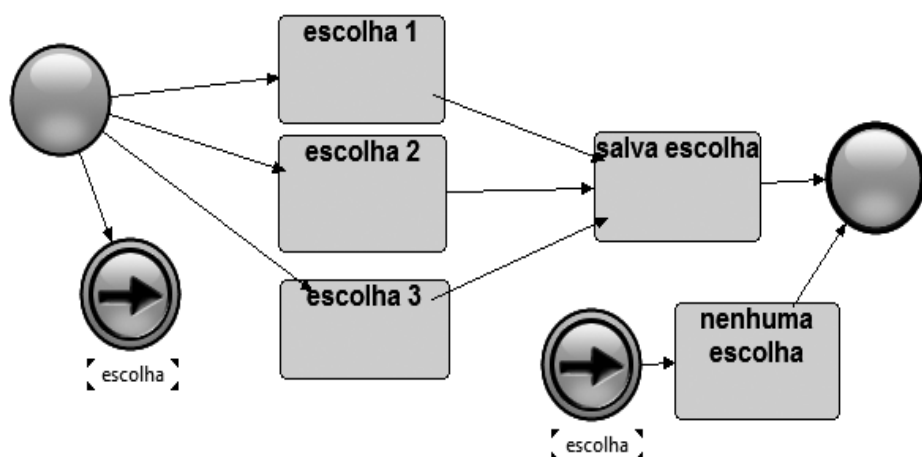
A forma como as soluções BPMS tratam esses erros e compensações é definida por sua implementação. Entretanto, a notação permite uma representação visual mais simples e representa erros de forma visual; erros que, de outra forma, teriam que ser representados como parte do fluxo, mas nada tem a ver com as regras de negócios. Se não tivéssemos a compensação e as exceções, teríamos um trecho como o da figura a seguir.



Nesse caso, prevemos o tratamento de um erro por meio de elementos condicionais; o que é ruim, pois os condicionais deveriam representar apenas decisões relativas à lógica de negócios. Quando se utilizam condicionais para representar tanto a lógica quanto os tratamentos de erros, tornam-se complexos os diagramas e, com frequência, de difícil compreensão. Quando se utilizam compensações e exceções, usam-se elementos diferentes para cada representação no diagrama.

## 11 - Conexão entre diagramas

Algumas vezes o diagrama se torna tão extenso que não conseguimos conectar certas partes, sem que as linhas fiquem cruzadas. Existe um mecanismo simples de link, que permite conectar elementos dentro de um diagrama, ou mesmo elementos entre dois diagramas distintos. Informamos que dois links estão conectados através de seu nome. O nome funciona como se fosse uma chave para os dois links. Um link pode ser representado como :



## 12 - Terminadores do processo

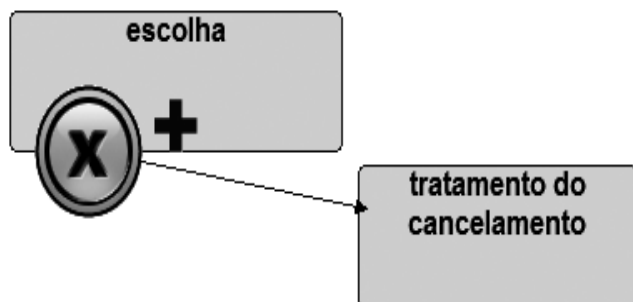
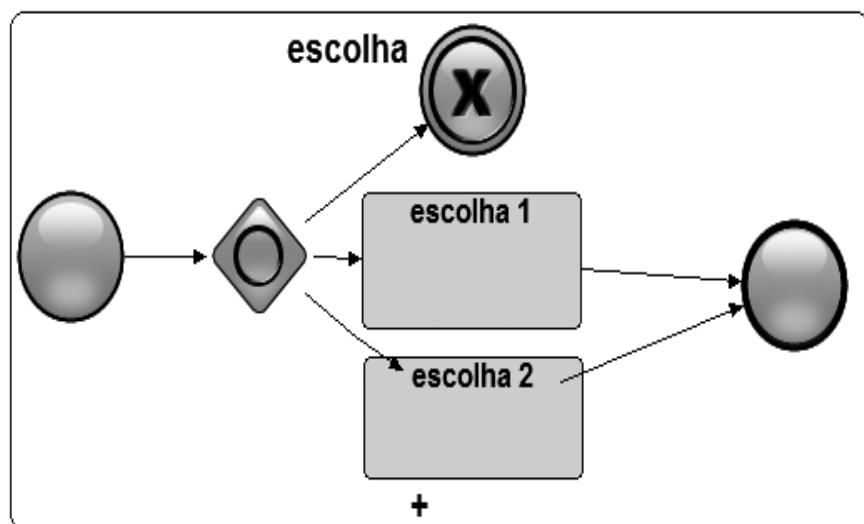
Existem atualmente três formas de finalizarmos um processo. As três representações possíveis são :



O primeiro tipo, com um círculo preenchido no interior, indica término incondicional. Isto significa que o processo deve ser terminado neste momento, sem compensação, tratamento de erros ou quaisquer outros tratamentos.







Se for um sub-processo, encerra imediatamente e retorna ao fluxo pai.

O término com um X no interior indica cancelamento da transação. Os mecanismos de compensação e erros trabalham em conjunto com o cancelamento. Normalmente o cancelamento é iniciado programaticamente, enquanto que a compensação e exceção são geradas por erros no processamento.



Supondo um sub-processo escolha, sendo que desdobrado se torna um fluxo como o da parte superior da figura. Caso a escolha do usuário for inválida, será gerado o cancel, que é capturado pelo evento cancel conectado à borda da atividade. Neste caso, se redireciona para um outro tratamento, onde a seleção não escolhida será tratada como se fosse uma exceção.

## 13 - Resumo dos tipos de eventos da BPMN

	Posição			na borda de uma atividade		no meio do fluxo	
	início	intermediário	final	mensagem	fluxo	mensagem	fluxo
geral				eventos genéricos			
mensagem				recebe evento	tratamento de exceção	recebe evento	envia ou recebe
timer				N/A	desvio temporizado	N/A	delay
exceção				N/A	aponta atividade de tratamento	N/A	gera exceção
cancel				N/A	aponta atividade cancelamento	N/A	força o desvio para o cancelamento
compensação				N/A	aponta atividade compensação	N/A	força o desvio para compensação
dados				N/A	exceção para valor de dado	N/A	inicia processo a partir de valor
link				conecta dois links, dentro de um mesmo fluxo ou em fluxos diferentes			
múltiplo				tratamento de eventos múltiplos			
terminate				Somente no final do fluxo, e significa término imediato			

## 14 - Gateways

Gateways são os mecanismos padronizados do BPMN para efetuarmos desvios. Os vários tipos de desvios, como AND (E), OR (OU) e XOR (OU exclusivo) são tratados com simbologias distintas pela notação. Assim como os eventos são representados por círculos, os gateways são representados por losangos.

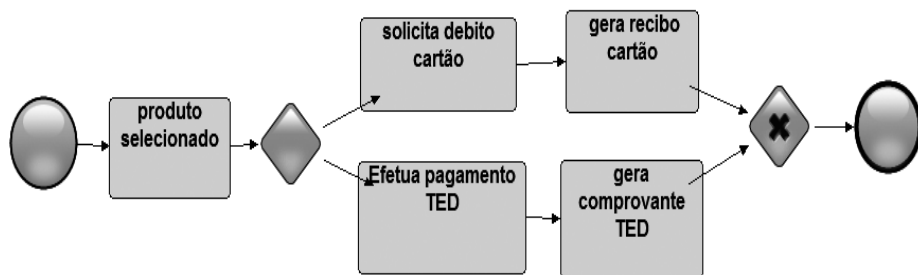
### 15 - Eventos do tipo XOR (OU exclusivo)

Os eventos do tipo XOR (OU exclusivo) podem ser representados de duas formas



Este tipo de gateway é o mais simples de se entender, pois ele representa o OU, onde o acesso a um dos caminhos é exclusivo. Ou seja, apenas um dos caminhos será seguido, não importando quantos caminhos existam para escolha.

Um exemplo deste tipo de representação poderia ser a escolha do tipo de pagamento de um produto :



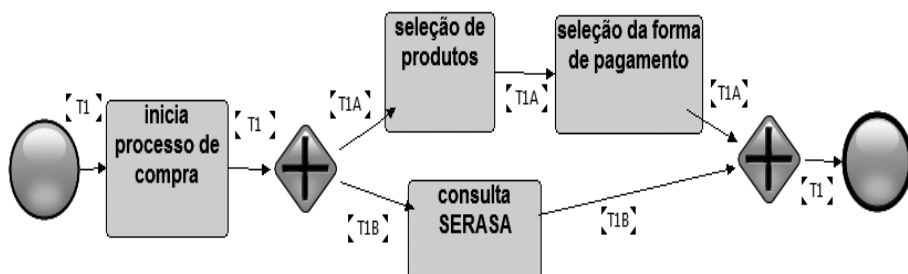


Neste caso, o pagamento será feito via transferência bancária ou por cartão de crédito. Qualquer dos caminhos tomados, haverá uma junção na frente, fazendo com que o fluxo continue após um dos caminhos.

Este caso é de fácil compreensão, pois apenas um dos caminhos é seguido. Existem situações onde mais de um caminho pode ser seguido (OR) ou mesmo todos os caminhos serão sempre seguidos (AND). Neste caso, para poder explicar melhor o conceito do que acontece nestes casos, a especificação utiliza o termo TOKEN. Precisamos entender este conceito para compreender os outros tipos de gateway.

## 16 - Token

Imagine que uma instância de fluxo de processos foi criada. Neste caso, a execução segue pelo caminho natural de execução, mas como existem gateways que permitem mais do que um caminho em paralelo, podemos ter dois caminhos sendo executados ao mesmo tempo, dentro de uma instância de processo. Veja este exemplo :



O processamento se inicia normalmente, com a criação da instância. Podemos chamar neste caso T1 a instância em execução. O problema é que quando passamos pelo AND (o gateway com sinal de mais dentro) dois caminhos são seguidos em paralelo, representados neste caso por T1A e T1B. Se estivéssemos nos referenciando a linguagens de programação, teríamos o processo em execução (T1) e duas "threads" executando em paralelo. É exatamente este o conceito de Token. Mas, se é equivalente à uma thread, porque não chamá-la diretamente por este nome ?

Bem, em primeiro lugar o termo thread diz respeito a uma linguagem de programação, e neste conceito os dois braços de execução (T1A e T1B) não precisam estar realmente executando ao mesmo tempo. A notação BPMN não força a que uma implementação de solução BPMS tenha todo o conceito de multi-thread embutido dentro dela. Mas a notação diz simplesmente o seguinte :

**"Uma vez criados mais de um token a partir de uma bifurcação, o processamento somente continua após todos os tokens chegarem até a junção".**

Esta frase abre margem para mais de uma interpretação, e em nenhum momento estamos dizendo que a execução está acontecendo em paralelo. Dissemos que os dois braços de execução precisam chegar ao ponto de encontro, para que o processamento continue após a junção do gateway. Desta forma, uma solução multi-threaded pode executar **"seleção de produtos"** e **"consulta serasa"** realmente em paralelo, o que seria interessante, mas também pode executar primeiro a atividade **"seleção de produtos"**, depois a atividade **"seleção de forma de pagamento"** e depois a atividade **"consulta serasa"**, para somente então continuar após a junção do gateway. Perceba que para qualquer das duas implementações, não há diferença aparente em termos de processamento, a não ser pela menor velocidade das atividades sendo executadas de forma seriada.

Paralelo ou não, necessitamos de um conceito para indicar esta independência criada pelo conceito de bifurcação e novamente dependência criada pela junção.

Com este conceito de Token em mente, podemos explorar os outros tipos de gateway existentes.

## **17 -Eventos AND (E)**

Este caso é exatamente o exemplo anterior. Quando se chega até a bifurcação do tipo AND, se seguem os dois caminhos, através de dois tokens criados. Não há processo de decisão, e todos os caminhos são seguidos. O objetivo de utilização do AND pode ser tentar otimizar o tempo de processamento, permitindo interações em paralelo pelos vários perfis de execução.

## **18 - Eventos OR (OU)**

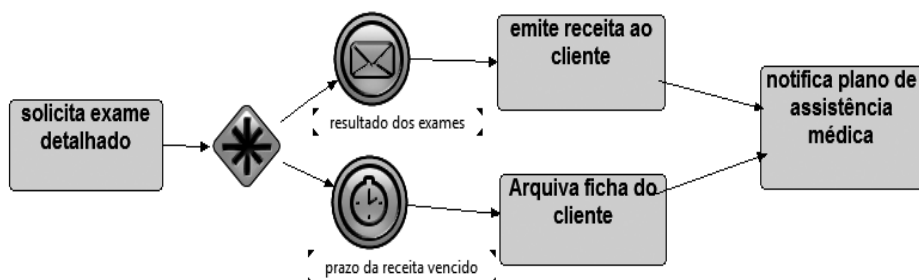
Os eventos do tipo OU permitem que se navegue por um ou mais caminhos durante o fluxo de execução. Ao contrário do OU exclusivo, pode-se seguir um ou mais dos caminhos. É necessário que ao menos um dos caminhos seja válido para navegação.

## **19 - Tratamento de eventos**

Outra necessidade decorrente da especificação BPEL são os eventos múltiplos. Imagine uma situação em que um processamento deve aguardar o acontecimento de um evento.

Na verdade, pode acontecer uma de várias possibilidades de evento. Esse evento pode ser o recebimento de uma mensagem ou um timer expirando depois de dois dias. Nesse caso, há duas possibilidades de eventos.

A primeira que acontecer continuará o processamento, e as outras opções são canceladas. Veja um exemplo na figura a seguir.



## 20 - Eventos múltiplos

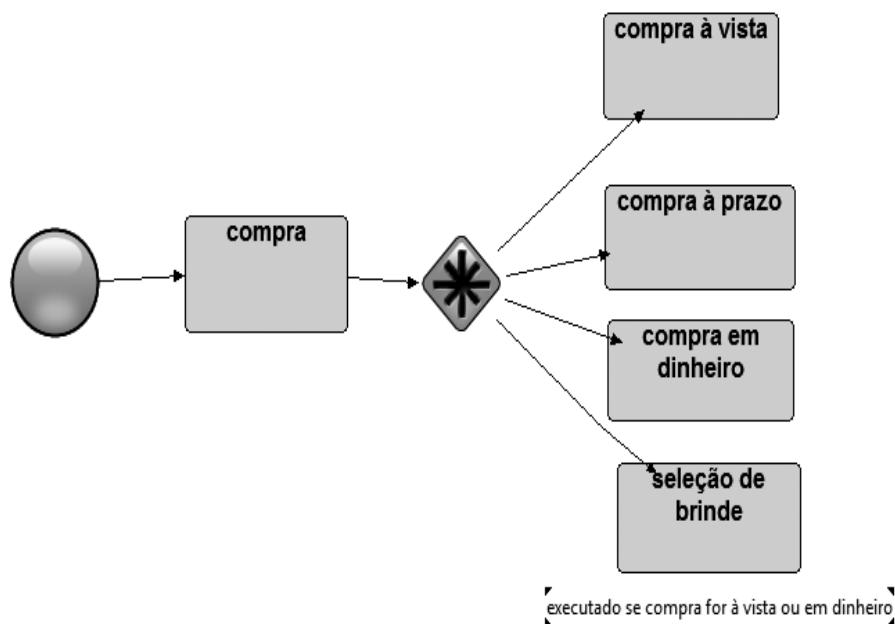
Nesse caso, o médico solicita um exame e duas são as possibilidades: o cliente pode fazer o exame solicitado e retornar ao médico, ou o cliente pode ignorar a solicitação e dirigir-se a outro médico. Caso o cliente faça o exame, o médico emitirá uma receita com base nos resultados. Caso não haja retorno, a ficha do cliente será arquivada. Nos dois casos, o processamento continua pela notificação do plano de assistência médica.

Vale salientar que, após adotar uma possibilidade, ignora-se a outra. Caso o resultado dos exames seja enviado ao médico, nunca ocorrerá o arquivamento da ficha do cliente. Esse tipo de estrutura é chamada de decisão complexa baseada em eventos e corresponde ao elemento "pick" da BPEL. Claramente a documentação indica que a implementação deste elemento foi no sentido de facilitar a migração de desenhos criados em BPEL.

## 21 - Eventos Complexos

Algumas vezes nenhum dos três tipos de gateways (OR, AND e XOR) é suficiente para representar uma situação. Os gateways testam uma única condição, mas algumas vezes precisamos testar mais de um dado para tomada da decisão.

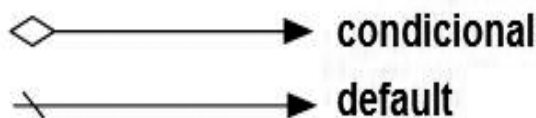
Outra necessidade é a mistura de um comparador comum (AND, OR ou XOR) com um comparador baseado em eventos. Neste caso, foi criado um coringa chamado “evento complexo”, onde cada uma de suas saídas pode efetuar testes distintos, e decisões podem ser tomadas de forma mais completa. Outra utilização para este tipo de elemento existe quando precisamos saber se uma determinada saída foi escolhida, em função de outras escolhas.



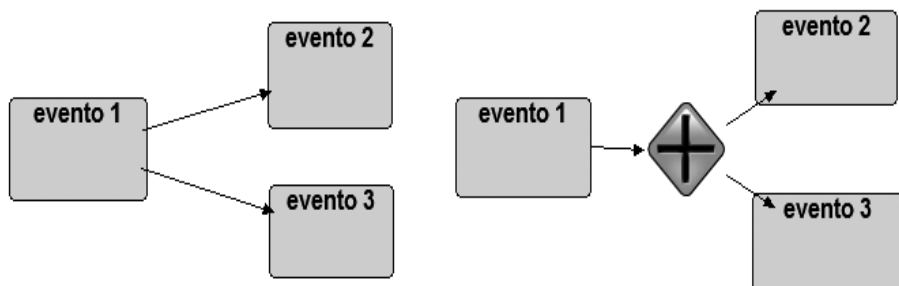
Neste caso, podemos ter compras à vista, em dinheiro ou à prazo. Caso a compra seja à vista ou em dinheiro, o caminho seleção de brinde também é executado. Caso a compra seja a prazo, seleção de brinde não é executado. Desta forma, uma saída pode tomar decisões em função de outras saídas. Os outros tipos de gateway não permitem este tipo de comportamento.

## 22 - Gateways diretamente em fluxos

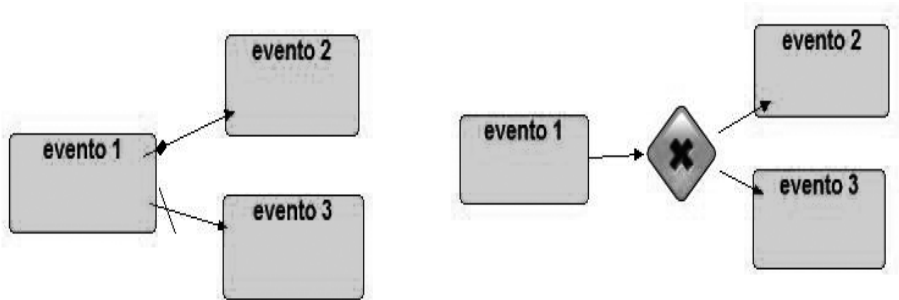
Embora obscuro, podemos representar gateways diretamente na saída das setas que saem das atividades. O diagrama se torna de mais difícil leitura, mas reduz a quantidade de gateways.



Quando a saída de um fluxo contém um losango, significa que uma decisão deve ser aplicada. Quando a seta contém um corte diagonal, significa que é a saída default. Se a seta não contiver nenhum destes dois elementos, significa que o caminho deve ser seguido. Se houver mais de um caminho, todos eles devem ser seguidos. Portanto, as duas representações a seguir são idênticas, em termos de BPMN.



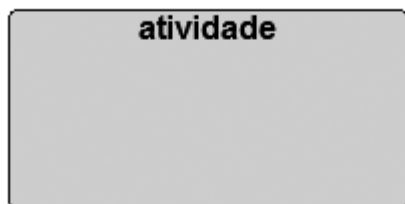
No primeiro caso, a saída da atividade segue por dois caminhos, o que significa que há uma bifurcação. No segundo, está representado através do gateway uma bifurcação. Para junções, o processo é o mesmo, ou seja, duas setas chegando a uma atividade representam a junção de um AND. Para elementos do tipo XOR, seria algo como :



No primeiro caso, o losango indica uma condição, que se não for respeitada, fará com que o fluxo siga pelo caminho default, para evento 3. Esta representação é similar ao desenho da direita, mas utilizando o gateway para XOR.

## 23 - Atividades e tarefas

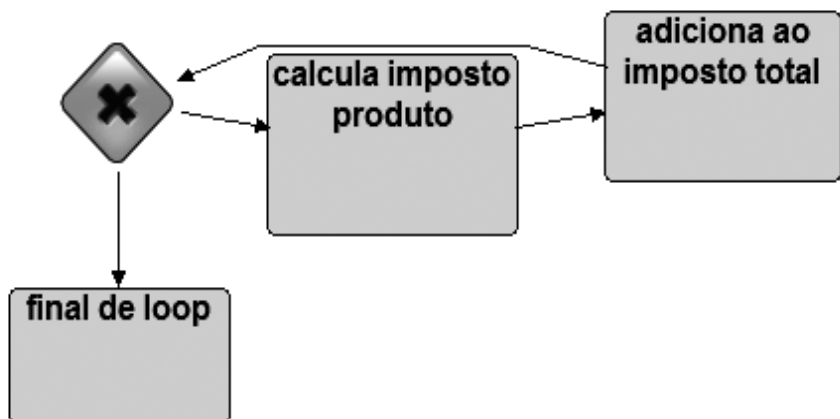
Uma atividade é uma unidade de trabalho em um processo. Pode significar uma interação com o usuário, ou algum processamento independente do perfil. Quando a atividade é tão isolada que não faz mais sentido subdivisões, ela é chamada de tarefa. Uma tarefa, portanto, é uma particularização do conceito de atividade. Da mesma forma, um sub-processo é um tipo de atividade onde sabidamente existe pelo menos um grau de detalhamento em seu interior. Se fôssemos dividir em temos de hierarquia, poderíamos pensar no processo como sendo a atividade de mais alto nível (algumas vezes chamado de macro-atividade). O processo é dividido em atividades menores, chamados sub-processos. Estes sub-processos vão sendo cada vez mais detalhados, até que chegamos em um nível onde não há mais detalhamento, este chamado de tarefa (task). De forma geral, qualquer tipo de atividade é representado por um quadrado com as bordas arredondadas.



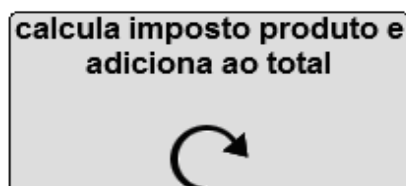
## 24 - Loop

Algumas vezes necessitamos executar repetidas vezes uma atividade, como por exemplo calcular o imposto para cada um dos produtos de uma lista de compra. Poderíamos utilizar um trecho de código para indicar este cálculo, algo como :

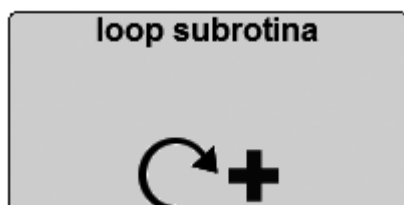
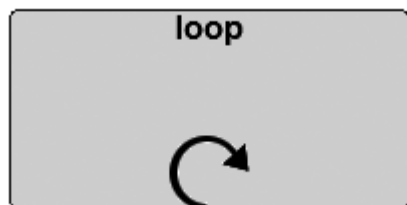




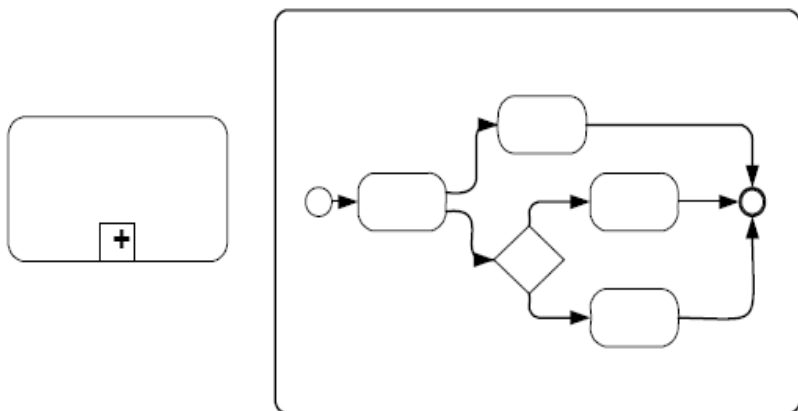
Basta colocar um condicional, normalmente um XOR, e uma condição para que o fluxo tenha uma saída. Para este caso, poderíamos representar através de uma atividade do tipo loop, de forma mais sintética.



Em muitas situações o loop irá executar uma seqüência de tarefas, portanto podemos utilizar um loop associado ao elemento de sub-rotina, que é representado por um sinal de soma na representação.

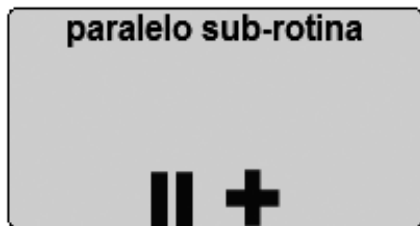


O padrão BPMN recomenda que uma sub-rotina tenha o sinal de soma na parte inferior, e caso seja "clicado" deveria se expandir, apresentando uma miniatura do diagrama que normalmente estaria em seu interior. Este comportamento não é obrigatório, mas parece ser a forma mais intuitiva de tratamento deste tipo de elemento.



## 25 - Múltipla instância

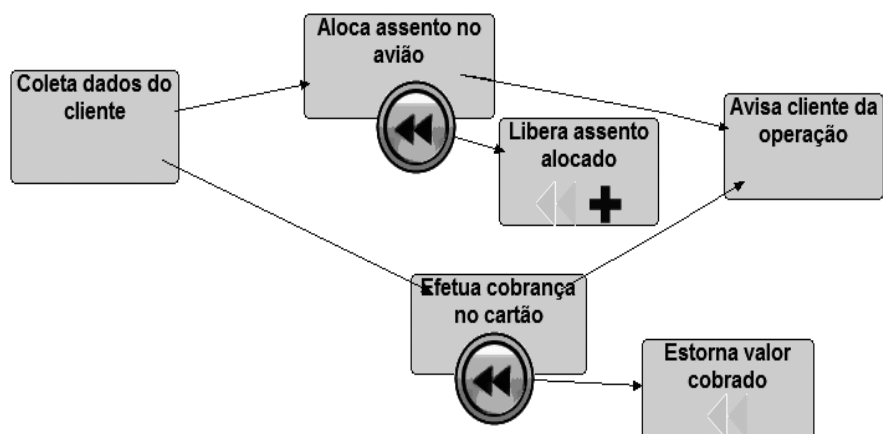
Outra forma de representação para o loop é a execução de múltiplas instâncias. Neste caso, a representação é de duas barras verticais dentro da atividade.



A diferença básica entre as duas representações é a forma como cada um dos elementos é tratado. No loop, os elementos são tratados um a um, mantendo sempre um único token ativo. No caso do processamento paralelo, para cada um dos elementos a ser tratado é gerado um novo token, fazendo com que cada um dos elementos tenha um tratamento independente, causando o processamento em paralelo nas implementações que suportem multi-thread. Na verdade, a escolha por um ou outro tipo de simbologia pode trazer impactos na implementação. Se necessitarmos sumarizar um valor, como no exemplo anterior de cálculo de imposto, o loop é recomendado. Caso os elementos possam ser tratados de forma totalmente independente (lembre-se de que cada token pode ser tratado como uma thread por uma linguagem de programação) podemos utilizar o paralelo.

## 26 - Compensação

A compensação é um tipo de atividade focada na execução de um rollback, ou retornar o estado de um processo a um estado anterior, para que o processo possa ser executado. Ele normalmente é utilizado em conjunto com o evento de compensação, que redireciona para esta atividade quando necessário.



Repare os dois triângulos dentro do evento de compensação, neste caso **“Libera assento alocado”** e **“Estorna valor cobrado”**. Eles indicam este tipo de atividade como uma compensação.

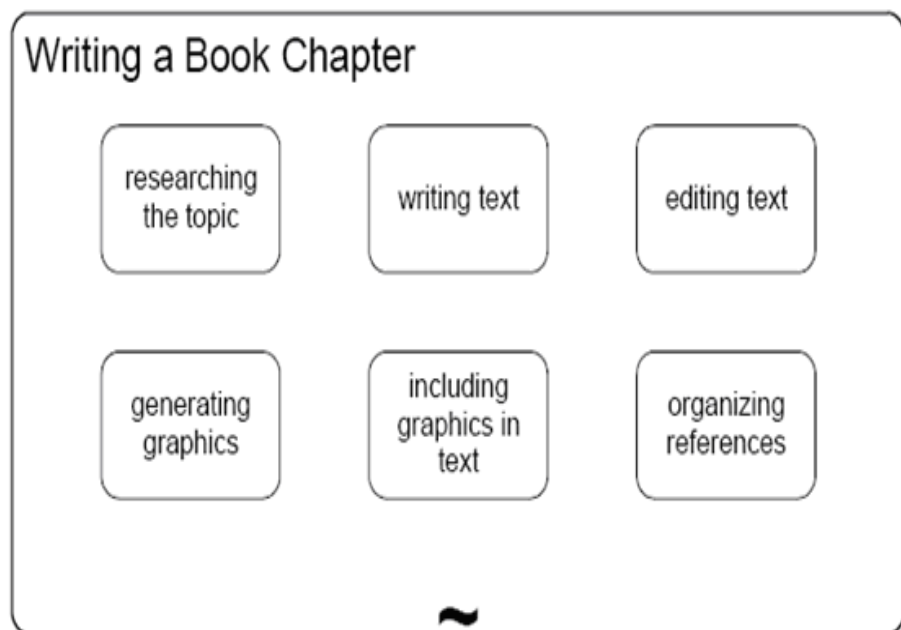
## 27 - Ad Hoc

Há uma grande discussão na internet sobre o BPMN ser capaz de representar situações de uso corriqueiro em termos de processos. O contra exemplo mais utilizado é aquele de um escritório de advocacia, lidando com um processo de cliente. Embora existam várias atividades a serem executadas, como cópias autenticadas de documentos, obtenção de assinaturas, agendamento de audiências, entre outros, muitas vezes não temos como garantir em que ordem estas atividades irão acontecer. O importante é que todas elas estejam finalizadas para que possamos passar para uma etapa seguinte. O mesmo pode ser aplicado à tarefa de escrever um livro. Não necessariamente há uma ordem para sua escrita, desde que todos os capítulos estejam prontos para sua publicação. Uma atividade do tipo Ad Hoc é identificada por um “til” dentro, mas as atividades em seu interior são soltas, ou seja, não estão conectadas. Considera-se o final da atividade Ad hoc quando todas as atividades em seu interior tiverem terminado.

A simbologia para este tipo de atividade é a seguinte :



Veja um exemplo de atividade do tipo Ad Hoc

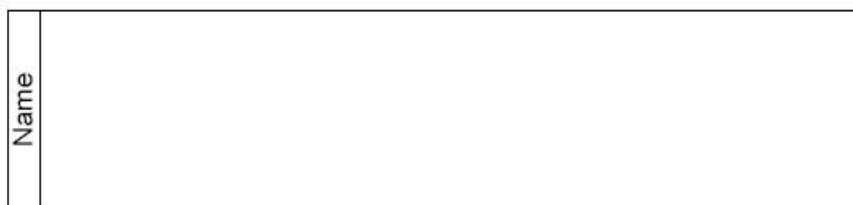


## 28 - Pools e Lanes

Pools são “compartimentos” onde os elementos do fluxo são acomodados, de forma a indicar que participante do processo ou um perfil está executando cada atividade. De forma geral, a pool ou lane não interfere nem define o que será feito, mas sim quem o fará. Está associado a mecanismos de segurança normalmente.

A especificação não define o tipo de elemento, que pode ser um departamento, perfil ou pessoa.

A representação recomendada é de uma caixa, com uma linha vertical separando os elementos de um título para o pool ou lane.

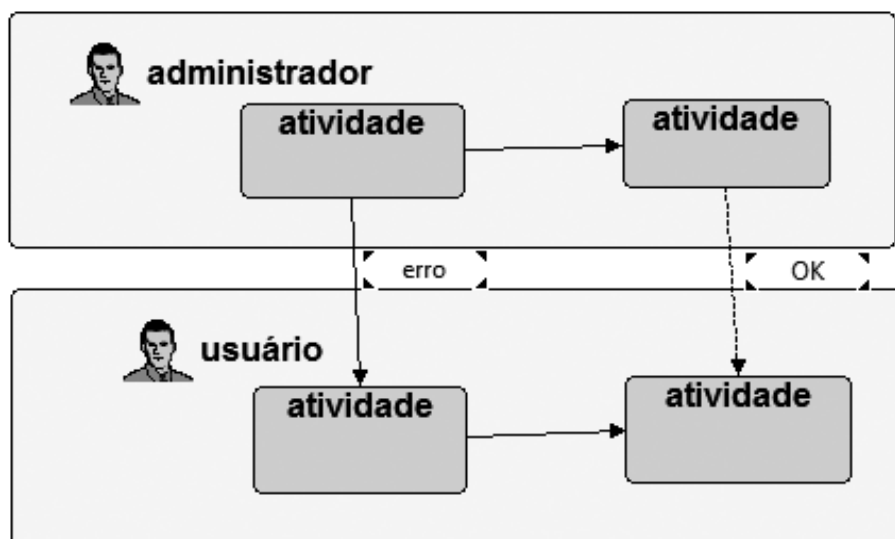


Uma pool é o elemento mais externo, ou seja, não se pode inserir pools dentro de pools. As lanes são elementos que são posicionados dentro de pools, para indicar mais de um perfil que colaboram para a execução de um processo.



Se os elementos internos de uma pool ou lane são representados, ele pode ser um diagrama privado ou de colaboração (leia o capítulo processos). Se por outro lado os elementos não estiverem visíveis, mas apenas como dois pools colaboram, chamamos de diagrama de colaboração.

Cada pool é considerado um fluxo isolado, e a única forma de elementos entre dois pools se comunicarem é através de mensagens. Ou seja, não é permitido fluxos entre dois elementos do tipo pool.

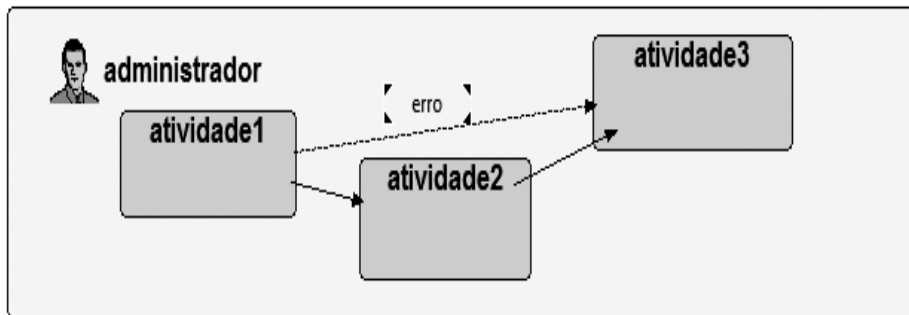


Podemos visualizar duas pools como duas instâncias de processos distintas, onde a única forma possível de interação é o envio de mensagens, onde esta mensagem enviada pode interferir no outro fluxo, caso este fluxo tenha sido preparado para o tratamento desta mensagem. No conceito de Tokens, existe ao menos um Token para cada pool, e estes Tokens nunca se encontram, ou seja, cada um permanece em sua instância de processo até o término de algum deles. Não há formas de se encerrar vários pools ao mesmo tempo.

O máximo que podemos modelar é enviar uma mensagem de um pool para outro, e modelar o segundo fluxo de forma que o processo também seja encerrado.

Em resumo, dois pools são como que dois programas totalmente distintos, onde a única forma de comunicação é através do envio de mensagens.

De forma similar, os mecanismos de mensagens foram criados para envio de informações entre dois processos distintos. Não faz sentido, e a especificação julga ilegal enviar mensagens dentro de uma mesma pool.

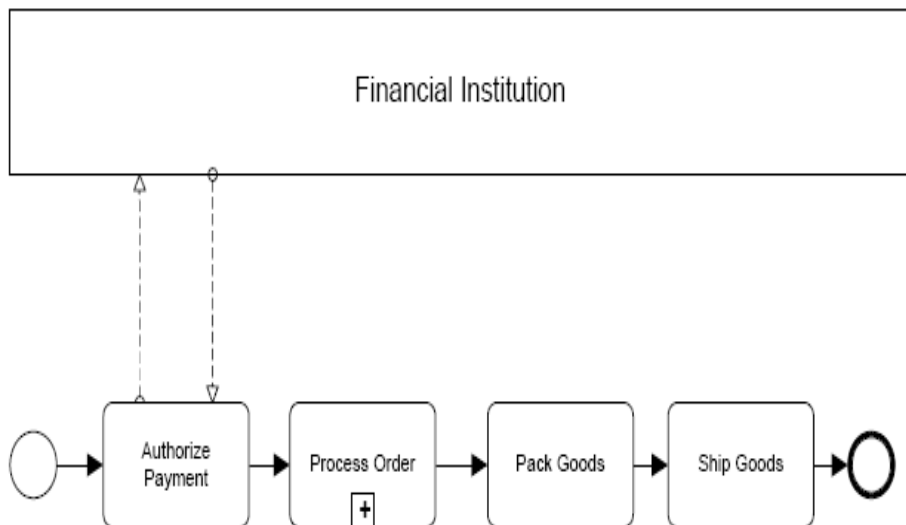


Esta restrição parece a mais fácil de compreender, se pensarmos em termos de tokens. No momento em que estamos na atividade1, existe apenas um token. O fluxo de processo irá continuar pela execução de atividade 2 e atividade 3 mantendo um único token. Mas, em atividade 1 estamos enviando uma mensagem para atividade 3, que somente será executada lá na frente.

Qual o sentido de enviar uma mensagem para uma atividade que não têm a capacidade de executá-la no momento ? Lembre-se de que o envio de mensagens não desvia o fluxo, e a atividade receptora somente pode tratá-la se estiver processando ou aguardando por ela. Portanto não faz sentido enviar mensagens dentro de uma mesma pool.



Se estamos trabalhando com pools abstratos (aqueles que não mostram os elementos internos) pode ser interessante mostrar os pontos de conexão entre duas pools, portanto é permitido enviar mensagens entre atividades de dois pools ou mesmo mensagens de um pool diretamente ao outro.



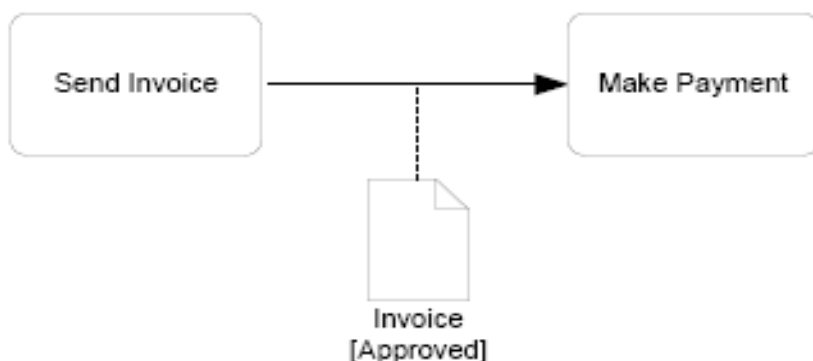
Neste exemplo, deve ficar claro que não é **“financial institution”** quem está enviando mensagens para o outro fluxo abaixo, mas sim uma de suas atividades internas. Como não estão visíveis quais são estas atividades internas, se moveu a representação da mensagem para a borda da pool.

## 29 - Artefatos

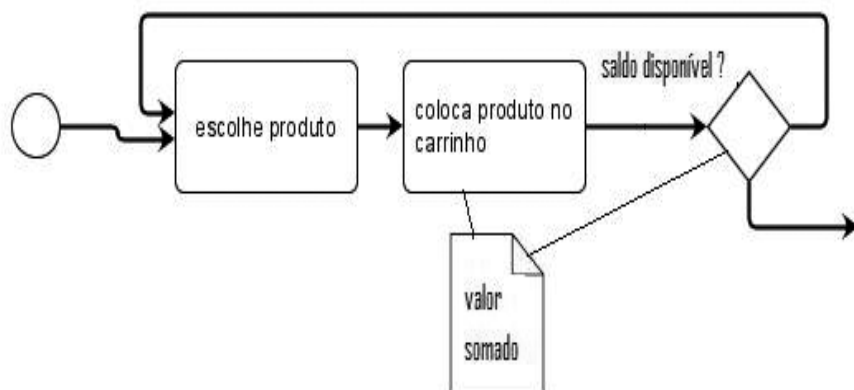
Artefatos são elementos extras que podem aparecer dentro do diagrama, mas que não alteram o fluxo nem executam tarefas. Eles servem como representações que irão aumentar a clareza do diagrama ou expor certos pontos importantes. A notação define um set reduzido, mas outros elementos poderiam ser acrescentados pelas implementações de ferramentas.

## 30 - Dados

Embora não seja preocupação do BPMN, algumas vezes desejamos tornar claro por quanto tempo e até onde um elemento de dados está sendo propagado. Este dado pode ser uma informação isolada ou mesmo um bloco de informações.

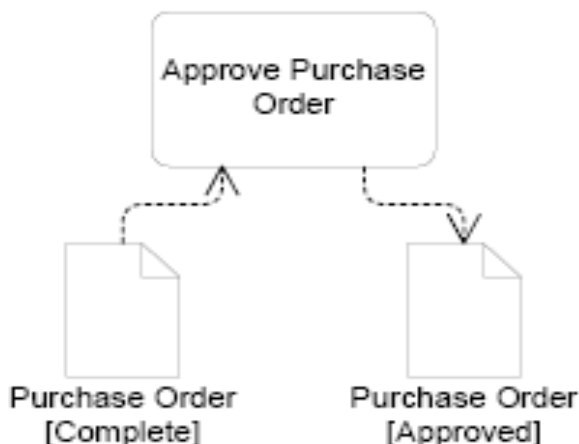


No exemplo a seguir, desejamos deixar claro que os valores estão sendo sumarizados, e poderão ser utilizados em um ponto a seguir no desenho dos processos.



A linha conectando o artefato de dados à atividade e ao condicional pode indicar que o elemento é válido durante aquele período de tempo, representando a longevidade da informação.

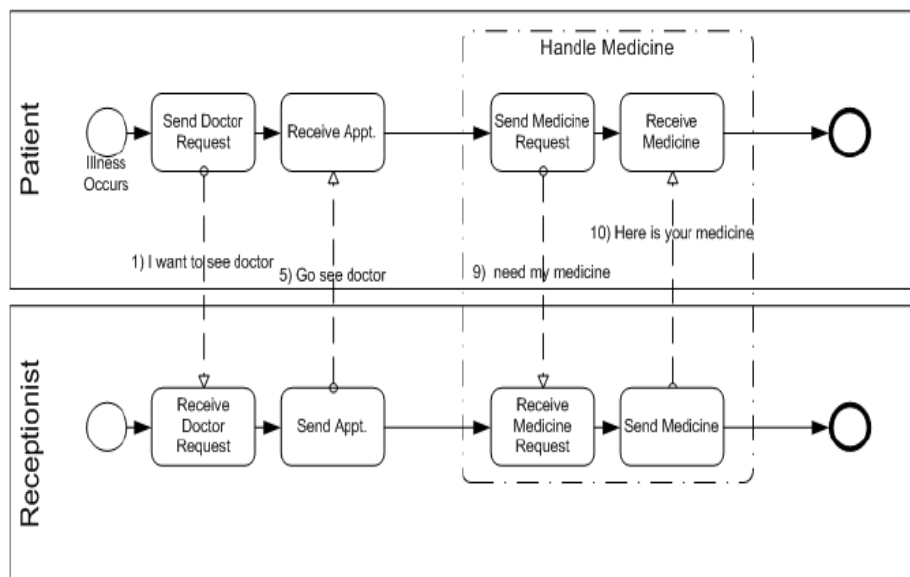
Também pode informar que um dado passou de um estado para outro, como no exemplo a seguir :



Na verdade, os elementos de dados podem funcionar como coringas que representam a informação, onde quer que ela esteja.

## 31 - Grupos

Grupos não afetam o fluxo de processos nem adicionam restrições. Da mesma forma que os elementos de dados, eles agrupam atividades e outros elementos de forma a tornar visíveis blocos importantes de operação. São meramente posicionais, e portanto é permitido que um grupo atravesse duas pools, como no exemplo a seguir.

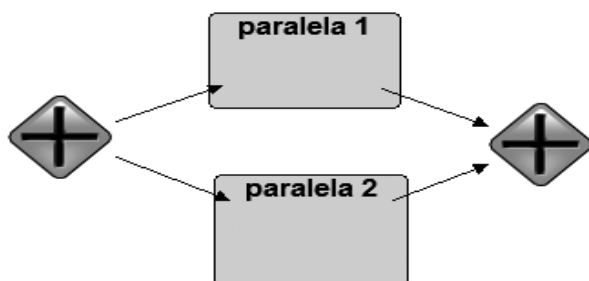


## **Parte II**

# **ERROS COMUNS NO DESENHO BPMN**

## 1 - Gateways de tipos diferentes na junção e bifurcação

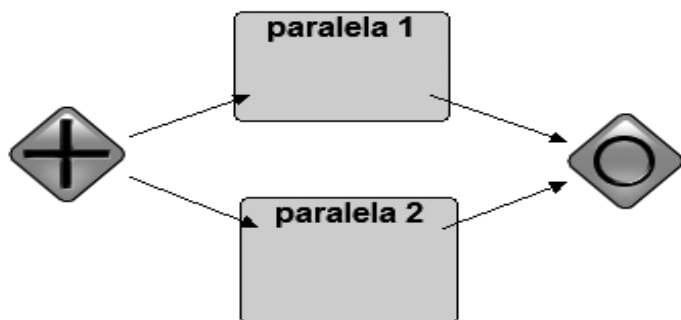
Normalmente utilizamos um mesmo tipo de bifurcação e junção, como no exemplo a seguir :



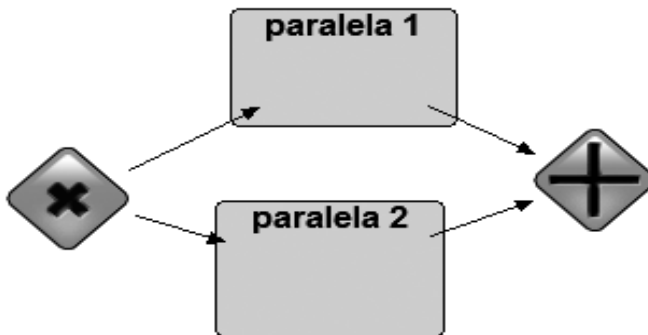
Isto garante que os tokens gerados serão sincronizados na saída. No exemplo anterior, foi criado um AND, que fez com que as atividades seguissem em paralelo. Na junção do fluxo será aguardado o término dos dois tokens, para que o processamento continue. Isto funciona desta forma porque a bifurcação de AND gera tantos tokens quantas forem as saídas, e a junção aguarda por todos os tokens gerados para continuar após o processamento.

Temos alguma flexibilidade nesta utilização, mas devemos utilizar com cuidado.

Uma possibilidade é a de adotarmos gateways de tipos diferentes, para junção e bifurcação. Ligeiramente diferente, o exemplo anterior ficaria :

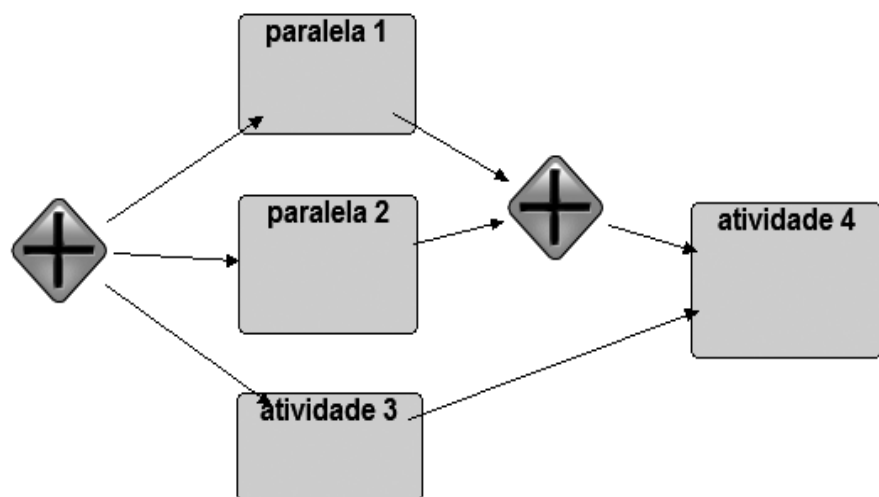


Neste caso, dois tokens serão gerados para “paralela 1” e “paralela 2”. Mas o token de junção é um OR, o que significa que a primeira das atividades que chegar ao fluxo fará com que o mesmo continue. Quando o segundo token chegar até a junção, será descartado, já que o fluxo já continuou assim que o primeiro Token foi detectado. Neste caso, pode não ser tão grave, a menos que as atividades após o gateway necessitem de informações ou dados gerados por alguma das duas atividades. Neste caso, como não sabemos qual irá terminar primeiro, corremos o risco de seguir adiante sem termos os dados consolidados para utilização. Mas, nesta mesma linha podemos cometer um erro grave, que poderá causar impacto muito sério na execução do fluxo criado.



Neste caso, apenas um token será gerado com absoluta certeza, já que a bifurcação é um XOR. Entretanto, colocamos um AND na junção, que irá aguardar pela chegada dos dois tokens, embora apenas um tenha sido criado. Acabamos de gerar um dead lock, que fará com que qualquer fluxo que passe por este trecho permaneça travado pela infinidade. Este termo (dead lock) é utilizado para um trecho em execução que causa um travamento, impossibilitando a continuidade da execução.

Outra modalidade de erro pode causar inconsistências no processamento, podendo ser até mais grave do que um dead lock.

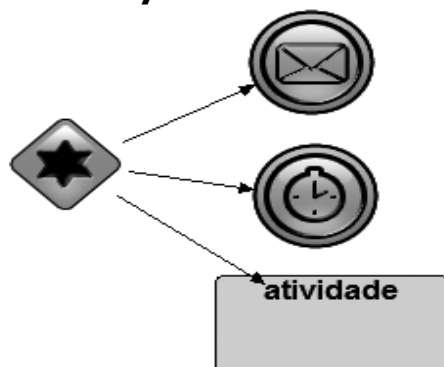


Neste caso, foi gerado um token, mas este não passou pela junção, chegando a um ponto mais distante do código. Não é difícil se criar fluxos com alguns níveis de gateways, tornando totalmente imprevisível o resultado de processamento dos mesmos.

Portanto, algumas regras podem salvar erros em um fluxo :

- 1) **Utilize bifurcações e junções do mesmo tipo**
- 2) **Garanta que uma atividade que passou por uma bifurcação continuou o fluxo passando por uma junção**

## 2 - Gateways baseados em eventos

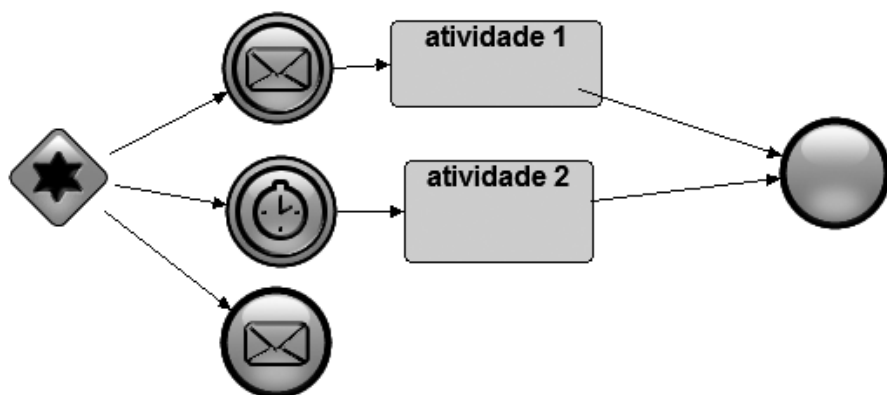




Os gateways baseados em eventos foram criados com o objetivo de permitir que um fluxo fique no aguardo de um evento, para continuar. Uma atividade não é um evento, e portanto esta representação não é permitida.

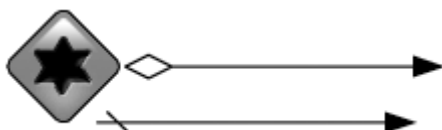
Outra particularidade sobre esta representação é que o gateway fica no aguardo de um dos eventos acontecer, e continua após a recepção do primeiro dos eventos. Todos os outros eventos são descartados após a execução deste caminho. Este tipo de condicional baseado em eventos é um XOR (OU exclusivo) pois somente um caminho é seguido.

Ao contrário dos outros gateways, este não necessita de uma junção para sincronizar os tokens. Veja um trecho de fluxo aparentemente inválido, mas correto sob ponto de vista da notação.

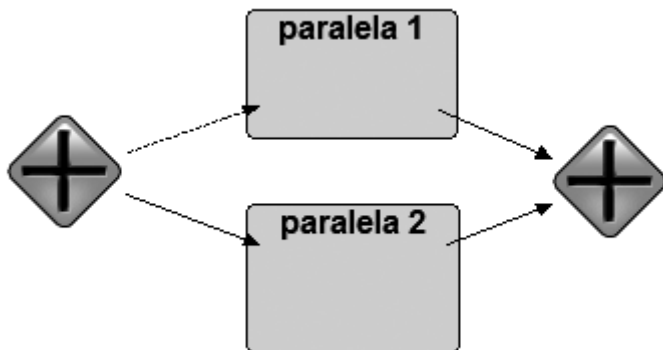


No caso anterior, caso o caminho seja o braço mais inferior, o processamento termina com o envio de uma mensagem. Se qualquer dos outros caminhos for seguido, o fluxo segue até o final do processamento.

### 3 - Utilização de elementos de fluxos com fluxos condicionais

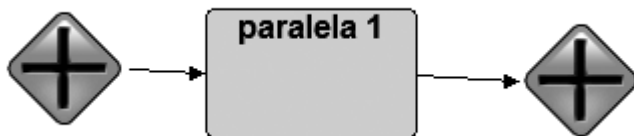


Não faz sentido utilizar as notações de condicional e default conectados à saída de um gateway, como no exemplo anterior. Ou utilizamos o gateway, ou utilizamos os fluxos diretamente na saída da atividade, utilizando desta forma a notação alternativa.



Fluxos não podem sair de gateways, como no exemplo anterior, nem mesmo chegar a ele. De uma forma geral, mensagens não podem ficar contidas dentro de pools. Todas as mensagens devem cruzar uma pool e atingir outro fluxo externo.

Embora pareça evidente, não podemos ter gateways com apenas uma saída. Eles são pontos de decisão, e devem fornecer alternativas ao caminho do fluxo.



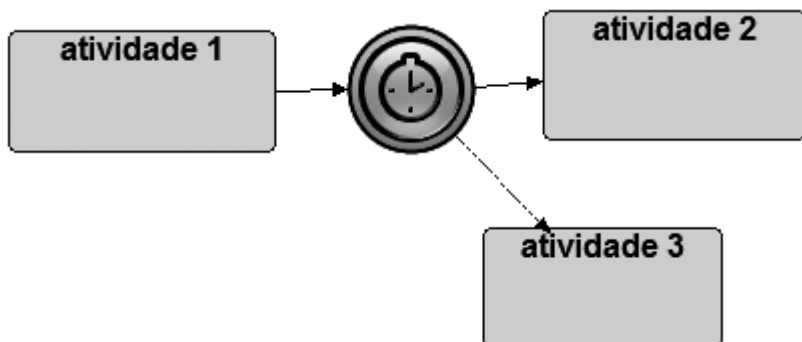
## 4 - Erros comuns no desenho BPMN – Eventos

A notação dita que os eventos de início apontam para o começo do fluxo, e eventos de finalização terminam um fluxo. Não é obrigatório que existam eventos de início e final, mas se existirem, devem aparecer aos pares. O seguinte exemplo portanto não é permitido :



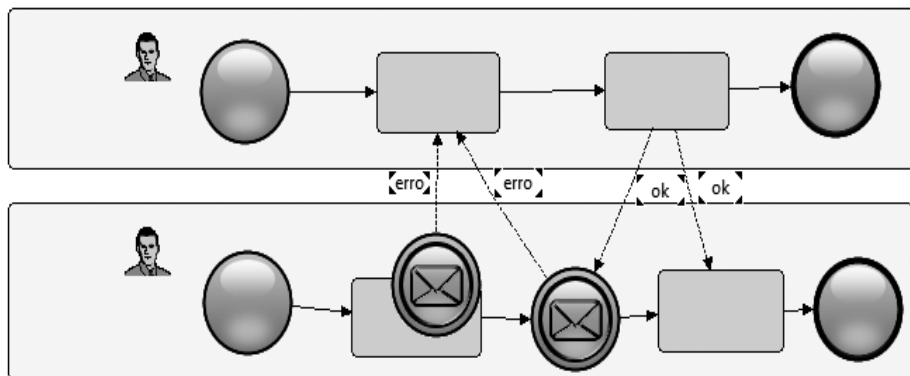
Por outro lado, podemos ter vários inícios para um fluxo, e se conveniente, podemos representar mais de um final. Quando um fluxo é iniciado, caso não existam os eventos de início todas as atividades que não tiverem uma seta chegando até ela são considerados pontos de início para o fluxo. Quando uma instância de fluxo é criada, são criados tantos tokens quantos forem os pontos de início do fluxo.

Outro erro comum relacionado aos eventos é que apenas mensagens ou atividades são geradores e consumidores de mensagens.



Neste caso, um timer não pode ser o gerador de uma mensagem.

Mensagens são geradas por atividades e podem chegar a outras atividades ou a elementos de evento de mensagens. Mensagens intermediárias não podem gerar mensagens. O exemplo a seguir está errado, portanto.



Em resumo, é correto enviar mensagens a partir de atividades, mas não podemos gerar mensagens a partir de eventos intermediários, estejam eles atachados à borda de uma atividade ou não. Como receptores, entretanto, esta restrição não existe.

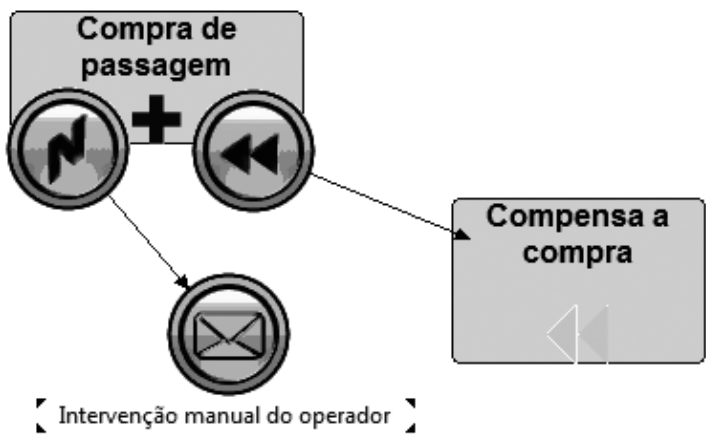
Um evento de timer não pode ser um finalizador de processo, afinal isto estaria significando que o processo estaria terminando “após um certo tempo”. Embora fosse bastante conveniente e útil, um evento de modificação de dados também não pode ser um finalizador.



Da mesma forma, os eventos de tratamento de erro, como exception, error e compensação não podem ser utilizados como eventos de inicialização.

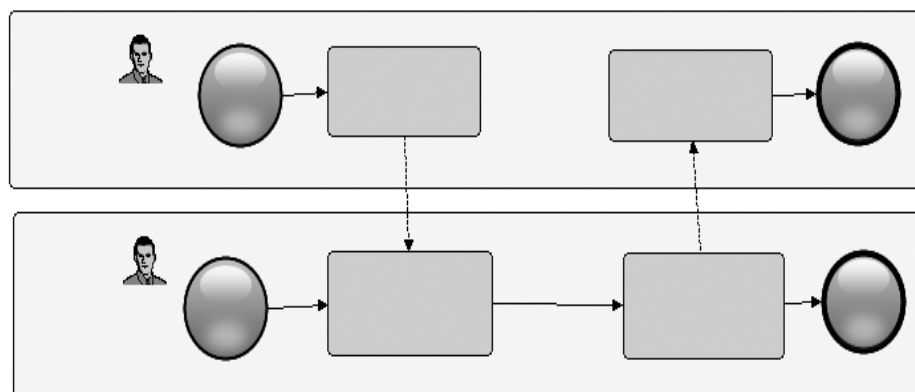


Por fim, os tratamentos de exceção são representados por setas de fluxo, e não de mensagens. Apenas lembrando que tratamentos de exceção são aqueles que saem das bordas de uma atividade, como no exemplo a seguir.



## 5-Fluxos, pools e lanes

Um fluxo deve se iniciar e continuar até o final do processamento. Não podem haver interrupções em sua representação.



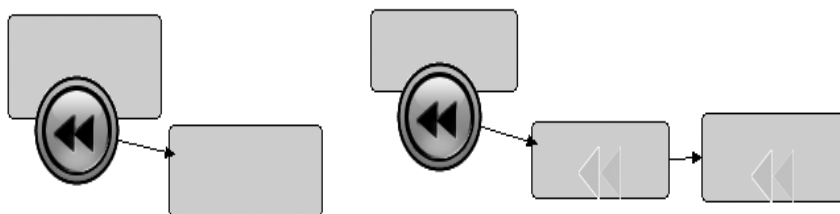
No caso anterior, não podemos “transferir” o controle para outro processo, e receber novamente o controle em outro ponto do fluxo, através de mensagens.

Vale repetir que entre pools a comunicação é feita entre mensagens. Dentro de um mesmo pool, a comunicação é feita através de fluxos de processo, que devem se iniciar e conduzir a sequência através das atividades até o final do processamento.

Dentro de uma pool, existe um único fluxo de processos. Quando um fluxo de processos é instanciado, todos os pontos de início recebem o processamento, criando-se tantos tokens quantos forem os pontos de entrada.

## 6 - Fluxos, pools e lanes

Uma exceção do tipo compensação deve desviar para uma atividade do tipo compensação, e apenas uma. As seguintes representações portanto não são válidas.



No primeiro caso, o desvio acontece para uma atividade comum, e não uma compensação. Uma exceção do tipo compensação deve apontar para uma atividade do tipo compensação.

No segundo caso, o desvio acontece para uma atividade que está ligada a outra compensação. Isto também não é permitido, e todo o tratamento da compensação deve ser feito em somente uma atividade (ou sub-atividade) do tipo compensação.

## **7 - Levantamento dos processos de negócios**

As técnicas para levantamento de processos de negócios não são muito diferentes da modelagem tradicional de sistemas orientados para objetos.

Normalmente a modelagem de processos aparece como uma etapa anterior ao desenvolvimento de um sistema, como forma de compreender os mecanismos do negócio em questão.

Muitas vezes eles nem mesmo se desdobram em sistemas, servindo apenas como desenhos para otimização dos processos da corporação.

Na maior parte do tempo, o levantamento de processos é feito através de entrevistas com usuários do sistema e normalmente geram artefatos que irão amadurecer para gerar documentos oficiais do sistema. O BPMN, neste caso, será um dos artefatos gerados para representação das informações. Perceba que a notação BPMN nem de longe é completa e uma série de outros diagramas e documentos são normalmente utilizados para entendimento do problema. Qualquer ferramenta de mercado, como Provision, MSVisio ou Enterprise Architect, Intalio e outros fornecem uma série de outros diagramas auxiliares para o desenho dos processos de negócios. Qualquer empresa fornecedora de uma ferramenta consultada irá apresentar sua “fórmula” para desenho de processos, quase sempre focada nas características de sua própria ferramenta.

Por isto iremos expor ao invés de uma metodologia, práticas que podem ser utilizadas de forma a ajudar no levantamento. Vamos propor uma série de idéias, e você pode manter em seu poder, como um canivete suíço, utilizando-as durante um levantamento, quando julgar mais conveniente.



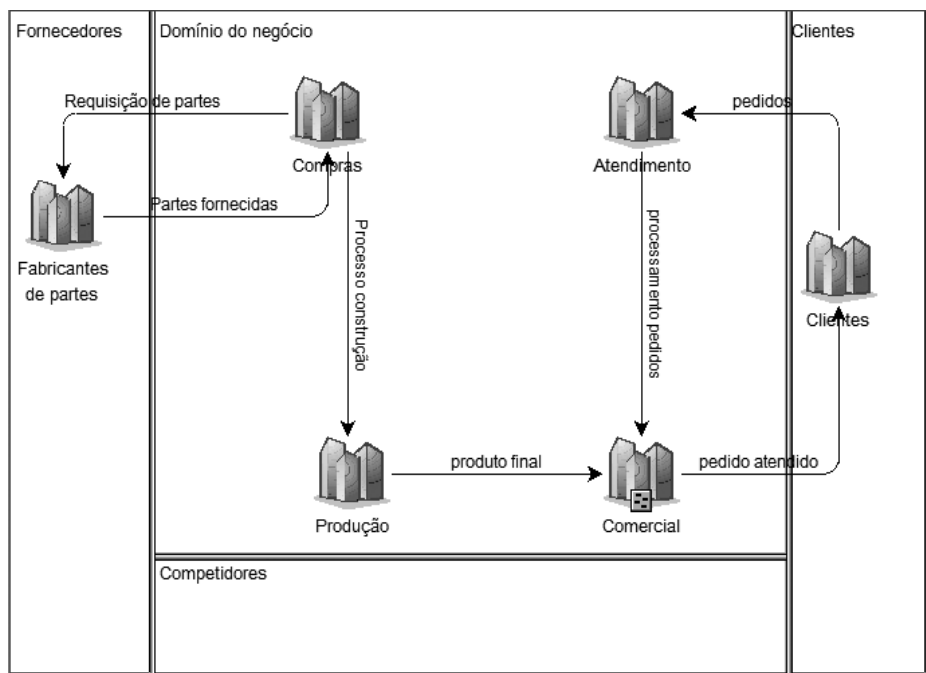
## **8 - Processos e Macro-processos**

Os desenhos de processos iniciam em uma camada superior de desenho, e vão sendo evoluídos (o correto seria detalhados) em níveis menores.

O nível mais alto que encontraremos são os macro-processos de uma empresa. O nível mais baixo que encontraremos seria uma atividade, que não pode ser subdividida.

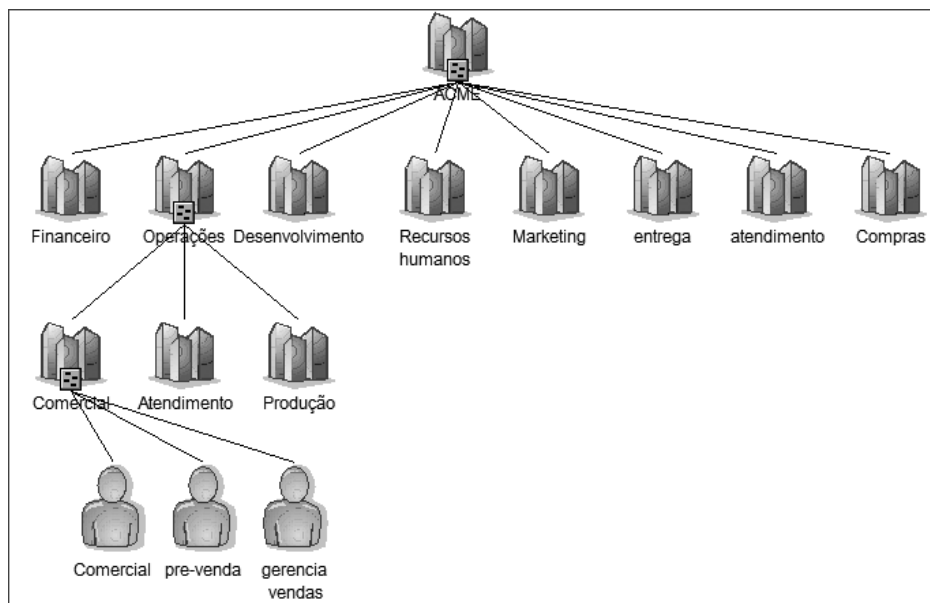
Imaginando que tenhamos iniciado neste momento nossas atividades, e não temos sequer os macro-processos da empresa. Especialistas costumam salientar que os macro-processos de uma empresa tendem a ser poucos, de um número que varia entre 7 a 15, mais ou menos. Não sou adepto de números mágicos, mas se uma empresa propõe a existência de 200 macro-processos, é provável que tenha ocorrido o que chamamos de decomposição funcional, ou seja, um macro processo foi detalhado em sub-processos.

Uma técnica que ajuda a descobrir os macro-processos de uma empresa é a criação de um diagrama de posicionamento. Ele posiciona a empresa, os clientes e parceiros dentro de áreas na tela, indicando o relacionamento entre cada uma destas áreas. Desta forma, cada uma das setas encontradas dentro deste tipo de diagrama tende a se tornar um macro-processo. Um exemplo deste tipo de diagrama pode ser observado a seguir :



Através deste tipo de diagrama, que pode ser feito em conjunto com os responsáveis pela companhia consegue-se identificar pontos de conexão importantes que irão gerar os macro-processos.

Ele pode ser desdobrado em outro que ao invés de mostrar os relacionamentos entre os blocos na empresa, apresenta como os departamentos estão hierarquizados.



Perceba que os dois tipos de diagramas anteriores são relacionados, e uma boa ferramenta de desenho garante que alterações em um sejam refletidos no outro.

Não devemos nos esquecer de que os processos são feitos e utilizados por pessoas. É sempre conveniente criarmos uma estrutura organizacional da empresa, que irá identificar os vários departamentos e perfis associados. Podemos partir do diagrama anterior, focado na hierarquia departamental, e ir evoluindo até que os vários perfis departamentais estejam visíveis. Há uma correlação direta entre estes perfis e com as pools e lanes que irão aparecer nos diagramas BPMN, quando já estivermos em uma fase mais adiantada.

## 9 - Os fluxos de processos

Quando estamos em uma sessão de entrevistas para levantamento de processos, muitas vezes não temos como forçar uma disciplina de forma a coletar as atividades de forma organizada.

Algumas vezes um entrevistado também efetua as mesmas atividades de forma diferente de outro no mesmo processo, e pode-se levar grande tempo discutindo o certo ou errado antes de avançarmos.

O mais simples nestes casos, é efetuar um "brain storm" levantando todas as atividades que devem ser efetuadas em um determinado processo.

Ao invés de conduzirmos uma discussão em termos já estruturados, as reuniões ficam mais leves se primeiro levantamos todas as atividades a serem executadas, e em um segundo momento as colocamos em um diagrama com a ordem de execução. Desta forma aliviamos a pressão imposta pela importância de um levantamento de fluxo, e a tendência a erros é minimizada. É importante que não exista a pressão para que as informações sejam fornecidas de forma organizada, mas sim manter o foco nas atividades componentes. Uma lista poderia ser algo como :

- **Seleciona o produto**
- **Coleta dados do cliente**
- **Sugere produtos relacionados**
- **Efetua cobrança no cartão**
- **Seleciona forma de retirada**
- **Preenche dados do cliente**
- **Assina protocolo de retirada**

Criar listas deste tipo são mais efetivas ao usuário do que já desenhar efetivamente o fluxo. Desenhar as várias atividades em papéis soltos e depois ir colando em uma folha em branco torna as reuniões mais efetivas e menos monótonas.

## **10 - “As is” e “To be”**

Costumamos dividir os desenhos de fluxos em duas fases. Em primeiro lugar se desenha o fluxo do processo como ele é, sem nenhuma alteração na forma com as atividades são processadas, por mais óbvia que seja a alteração. Costumamos chamar esta fase de “as is”. Em um segundo momento, com o fluxo completamente desenhado, se atua em sua melhoria, tornando-o mais eficiente.

Desta forma teremos o processo desenhado antes e depois das melhorias. Com estes dois desenhos em mãos, podemos nos utilizar de ferramentas de simulação que irão indicar o ganho de qualidade obtido com o redesenho.

Principalmente os especialistas focados em análise e levantamento, quanto maior o conhecimento do negócio, mais rapidamente vislumbram pontos de processamento desnecessários ou duplicados. Isto se deve à vivência no ramo.

É muito importante durante o levantamento de processos que foquemos em como o processo está atualmente ( as is), deixando o “to be” para um momento posterior. Muitas vezes o usuário do fluxo percebe o erro que está cometendo no momento deste desenho, e já tenta guiar o desenho de forma a contemplar esta melhora.

É importante que isto não aconteça, senão qualquer informação relativa à melhoria irá se perder e análises futuras nunca conseguirão ter sucesso no diagnóstico. Isso é importante pois irá gerar relatórios de melhorias de processo, indicando a economia ou aumento de lucro obtido.

O objetivo deste redesenho é a melhoria, e o usuário não deve se sentir culpado por não ter percebido isto antes, pois afinal ele estava tão focado na execução de seu fluxo diário que dificilmente teria como abstrair e imaginar melhores formas de realizar a tarefa.

## **11- Use Cases e processos**

Principalmente no momento em que a área de negócios se encontra com a TI, para a transferência de informações que irá permitir a implementação de um sistema baseado no fluxos de processos, surge uma dúvida muito pouco discutida que é :

Como uma atividade BPMN está relacionada com elementos de análise tradicional, como um Use Case, por exemplo ?

Neste caso, estamos nos referenciando aos Use Cases da modelagem orientada para objetos.

Na orientação para objetos nosso foco primário é manter dentro dos limites de um sistema, normalmente departamental.

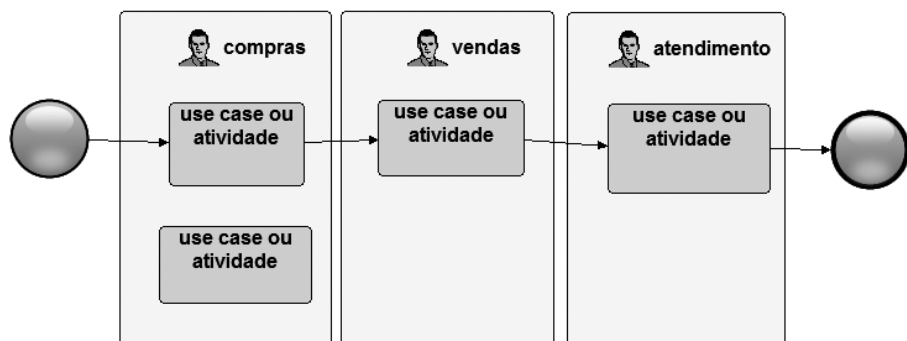
Quaisquer interfaces externas costumam ser ignoradas, e colocadas como caixinhas a serem implementadas no futuro. Com um exemplo prático, quando iniciamos a modelagem do sistema de vendas, raramente nos preocupamos com o sistema de estoque.

Isto é o oposto do que normalmente acontece quando uma empresa se organiza em termos de processos. Para uma modelagem em torno de processos, pouco importa a qual departamento uma atividade está relacionada. O que importa é a continuidade do fluxo, passando por vários departamentos da empresa até seu término.

Um fluxo de processos pode se iniciar como um processo de vendas na loja e passar pelo controle de estoque em algum momento, efetivando o pedido e a reserva do produto.

Por isto se costuma dizer que os sistemas atuais são verticalizados (ou seja, giram em torno de um departamento), enquanto que os processos permeiam a empresa na horizontal (ultrapassam vários departamentos).

Frequentemente se verifica este tipo de desenho :



Os mesmos blocos de mais alto nível dos que compõem os sistemas orientados para objetos, chamados de use cases, podem ser as atividades de mais alto nível que compõem os processos. Quando se decompõe uma atividade, ela se torna um sub-processo que na modelagem por processos irá se tornar um diagrama BPMN. Na orientação para objetos, um use case quando “decomposto” pode se tornar um diagrama de atividades, que é muito similar ao diagrama BPMN. Veja dois exemplos na introdução deste livro.

Em resumo, um use case é uma atividade de mais alto nível. Podemos utilizar as atividades de um fluxo de processos como insumo para detalhamento dos use cases, gerando os artefatos necessários para a UML tradicional, como diagramas de classes e de sequências.

E se um Use Case é essencialmente uma Atividade de mais alto nível, podemos utilizar os templates de detalhamento de use cases que já utilizávamos na UML para descrever como cada atividade se comporta. Embora um diagrama BPMN seja muito mais detalhado do que um diagrama de atividades, ainda não chegamos a um ponto onde possamos omitir o detalhamento da atividade.

Utilizando um template para definição de Use Case já na modelagem de processos de negócios utilizamos um mesmo artefato padronizado até os níveis mais baixos, facilitando no momento posterior da implementação do sistema.

## **12 - Conclusões**

Os objetivos primários da modelagem de processos de negócios é obter entendimentos de como a empresa funciona internamente.

Os artefatos discutidos, como organogramas, diagramas de posicionamento, fluxos de processos são formas visuais de compreender o que as pessoas fazem no dia a dia de suas atividades, e servem como ponto de partida para outros estudos, como melhorias nos processos e estimativa de custos por atividades, entre outros.

Durante este entendimento é importante formentar discussão, “levantando a poeira do tapete”, pois somente desta forma poderemos de forma correta compreender os processos corporativos.

É errado efetuarmos este levantamento apenas com o gerente da área, por exemplo, mas se for necessário, por falta de tempo dos outros envolvidos, devemos ao menos garantir que todos estão coniventes com esta visão.

Muitas vezes os funcionários fazem suas atividades de forma diferente do que a gerência imagina, e vice versa.

A modelagem de processos de negócios é importante pois nos faz compreender a empresa e os mecanismos que foram utilizados para que funcione. É fundamental não apenas como etapa pré desenvolvimento de um sistema, como também para adequação de uma solução de mercado, como um ERP, CRM, etc.



## 13 - BPEL e BPMN

Se hoje em dia temos um consenso em termos de notação para desenhos de processos, que é o BPMN, o mais próximo que temos atualmente de uma notação padronizada para execução deste processo é o BPEL (ou BPEL4WS).

A sigla é o anagrama de Business Process Execution Language, e foi criada como uma forma de permitir a orquestração de serviços Web (Web Services). Como uma linguagem para execução de serviços, ela é formal e não permite ambigüidades.

Uma das idéias no mercado é que BPEL seja utilizada na execução de fluxos de atividades. Nem todos os fabricantes a adotam em sua plenitude, e alguns incorporaram modificações para torná-la compatível com os recursos de seus produtos. Mas sua força é inegável, já que até mesmo a especificação BPMN na release 1.0 apresentava como fazer mapeamentos para o BPML, outro padrão também mantido pela mesma organização, e já na versão final do documento o capítulo foi substituído pelo mapeamento em BPEL.

Em essência, o BPEL é um arquivo XML. Dentro dele vários trechos XML vão indicando o que fazer durante o processamento de um fluxo, definindo e armazenando variáveis. O BPEL por si só não é capaz de executar muita coisa, delegando a responsabilidade para web services.

Se estivéssemos comparando o BPEL com um programa tradicional, poderíamos dizer que cada subrotina no programa seria um serviço, e o programa principal, que chama em seqüência cada rotina seria o BPEL. A desvantagem de utilizar BPEL é óbvia, já que interpretar um arquivo XML é mais lento do que executar um programa. Mas porque o mercado fala tanto sobre ele ?

Para uma solução BPMS, o BPEL traz vantagens. É muito comum ouvirmos sobre “realinhamento dos processos de negócios”, que significa reajustar um processo ou otimiza-lo para melhor execução.

Na maioria das situações, este reajuste é feito apenas na ordem em que os serviços são chamados. Se a execução deste fluxo está em um código, quando é necessário reajuste temos que alterar o código que executa o fluxo.

Se utilizamos um arquivo XML (BPEL), as alterações na ordem de execução são mais simples, já que não precisamos recompilar códigos, e ainda podemos manter duas versões no sistema, uma antiga, com os fluxos com a execução já iniciada, e um novo, com a nova forma de execução para os novos processos iniciados.

Desta forma, testes com o sistema em execução podem ser feitos com prejuízo mínimo aos usuários, pois os que haviam iniciado no fluxo antigo continuam até o final do processamento. Isto reduz o impacto nas alterações, pois não afeta o que já está em execução. Se utilizássemos códigos para fazer o mesmo processamento, seria muito complicado mantermos duas versões do código em execução na máquina. Delegando para o XML, o próprio engine BPEL pode fazer a distinção entre novo e velho.

Para permitir a execução e até mesmo passagem de parâmetros entre as chamadas, o BPEL possui um conjunto de elementos para lidar com situações comuns de programação, como declaração de variáveis, atribuição e cópia de valores e chamadas a serviços.

<b>&lt;invoke&gt;</b>	<b>- Efetua uma chamada a um serviço</b>
<b>&lt;receive&gt;</b>	<b>- Fica no aguardo de um serviço chamado</b>
<b>&lt;assign&gt;</b>	<b>- atribui uma variável</b>
<b>&lt;reply&gt;</b>	<b>- Responde com uma chamada</b>
<b>&lt;throw&gt;</b>	<b>- tratamento de exceções</b>
<b>&lt;wait&gt;</b>	<b>- Fica no aguardo</b>

Mas também temos elementos que controlam o fluxo, como ifs,elses e outros tipos de elementos comuns a linguagens de programação tradicional.

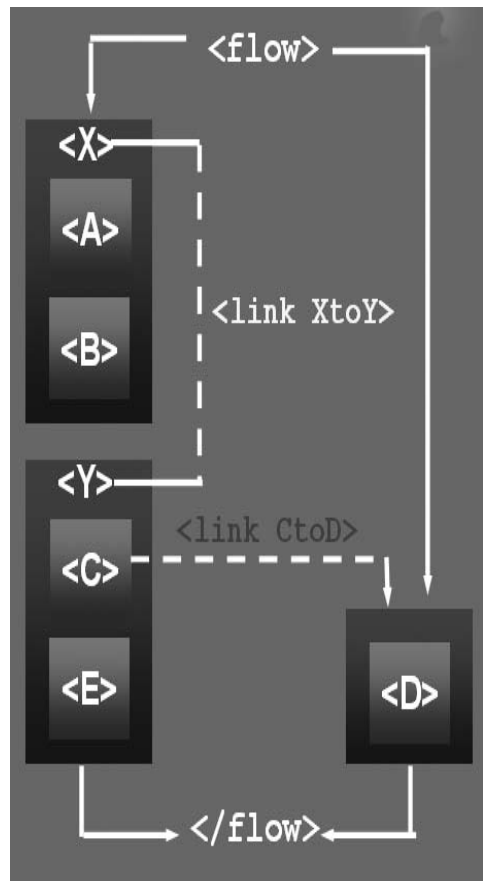
- <sequence>** - Executa em sequência
- <switch>** - Escolhe um caminho para execução
- <pick>** - Fica no aguardo de um evento
- <flow>** - Executa processamentos em paralelo
- <while>** - Executa um loop
- <scope>** - Define um escopo para atribuição ou leitura de uma variável

Um arquivo BPEL normalmente tem uma estrutura do tipo :

```

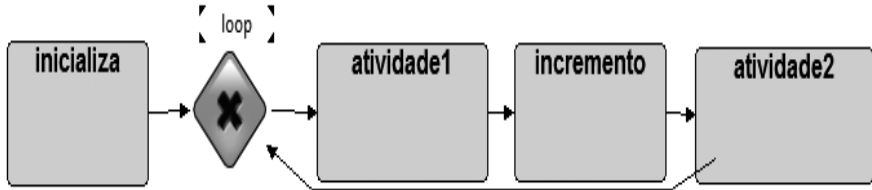
<flow>
<links>
  <link name="XtoY"/>
  <link name="CtoD"/>
</links>
<sequence name="X">
  <source
linkName="XtoY"/>
    <invoke name="A" .../>
    <invoke name="B" .../>
  </sequence>
<sequence name="Y">
  <target
linkName="XtoY"/>
    <receive name="C"/>
  <source
linkName="CtoD"/>
    </receive>
    <invoke name="E" .../>
  </sequence>
  <invoke partnerLink="D">
    <target linkName="CtoD"/>
  </invoke>
</flow>

```



## 14 - O Mapeamento BPMN para o BPEL

É necessária uma certa criatividade para mapear elementos do BPMN para o BPEL. Podemos apresentar um trecho de código e indicar os impactos no mapeamento :



Este trecho poderia ser mapeado como :

```
<variable name="contador" messageType="loopCounterMes  
sage" />  
<variable name="limite" messageType="forEachCounterMes  
sage" />  
<sequence>  
  <assign name="init_contador">  
    <copy>  
      <from expression="0"/>  
      <to variable="contador" part="loopCounter" />  
    </copy>  
  </assign>  
  <assign name="init_limite">  
    <copy>  
      <from expression="10"/>  
      <to variable="limite" part="forEachCount" />  
    </copy>  
  </assign>  
  <while condition=" bpws:getVariableData(contador,loopC  
ounter) <= bpws:getVariableData(limite,forEachCount)">  
    <sequence>  
      <invoke name="atividade1" ... >  
      <assign name="incremento">
```

```

<copy>
  <from expression="bpws:getVariableData(contador,loopCount)+1"/>
  <to variable="contador" part="loopCounter" />
</copy>
</assign>
<invoke name="atividade2" ... >
</sequence>
</while>
</sequence>

```

Em primeiro lugar, vale salientar que foi criada uma atividade chamada incremento, com o objetivo de manter o valor durante a execução do processo. Os dois invokes, que farão as chamadas aos serviços estão envolvendo um assign, que faz o incremento. Tudo isto está dentro de um while, que garante a execução em looping.

Foi inserida dentro do BPEL uma linguagem de expressão, o que permite a comparação de valores, como em "**bpws:getVariableData(contador,loopCounter)**".

Bem parece que temos tudo o que precisamos, certo ? Bem, nem tanto assim.

Primeiro a atividade incremento, quando foi mapeada por um analista de processos provavelmente não foi colocada. Ela apareceu no momento do mapeamento para BPEL, permitindo a geração da "rotina" de incremento.

**Isto faz com que um fluxo desenhado pelo analista raramente consiga ser mapeado diretamente para o BPEL, sendo necessários reajustes, neste caso exemplificado com a inserção de uma atividade para permitir a contagem.**

Outra particularidade oculta, mas a ser considerada é que o BPEL não têm o conceito de pools e lanes. Portanto não há perfis e se assume que um fluxo será executado para um perfil específico. Isto pode ser um problema para um desenho mais complexo, e talvez cada lane tenha que ser mapeada

Como os fabricantes estão lidando com estas deficiências ? Como dissemos anteriormente, muitos estão acrescentando elementos ao padrão, de forma a contornar estas inconveniências. Isto deve gerar nossa preocupação, pois podem surgir BPELs e BPELs, cada um com suas particularidades.

Como as empresas lidam com isto em termos de equipe ? Atualmente a abordagem comum é que os analistas que fazem o levantamento dos processos não se preocupam muito com a notação (alguns nem mesmo utilizam o BPMN).

No momento em que o fluxo for implementado em uma solução BPMS, será redesenhado para a ferramenta adotada, seja BPEL ou outra qualquer.

Isto não é bom, pois pode causar distorções na transcrição, além de dificultar o processo de implantação e praticamente inviabilizar o realinhamento rápido do negócio.

Também vale salientar que o invoke no BPEL faz uma chamada à um Web Service. Mas e se tivermos interações com os usuários ?

Talvez a mais notada proposta no mercado seja o BPEL4PEOPLE, proposta pela IBM e SAP, como alteração no BPEL de forma a permitir interações entre pessoas. O documento é público, e alguns fabricantes estão suportando o padrão. É necessário deixar claro que nem todos os engines BPEL suportam BPEL4PEOPLE, o que faz com que um processo modelado em BPEL nem sempre possa ser executado em qualquer engine.

Acho que vale salientar que os elementos do BPEL4PEOPLE são muito diferentes dos elementos BPEL tradicional, e ajustes profundos nos engines de execução são necessários

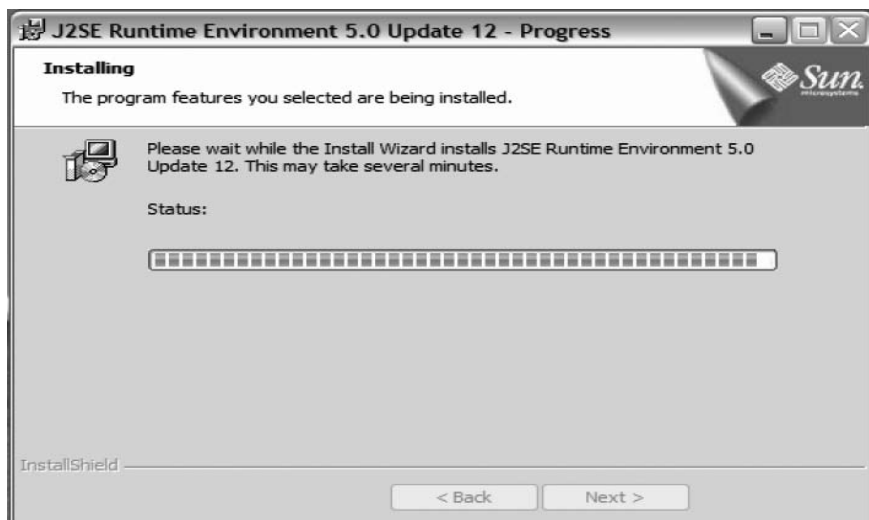
## 15 - Editor BPMN da revista PortalBPM

Você pode localizar este editor BPMN no site da revista PortalBPM ([www.portalbpm.com.br](http://www.portalbpm.com.br)), na seção de downloads. Para que mais uma ferramenta de desenho ?

Bem, esta ferramenta é distribuída gratuitamente e contém todos os símbolos da notação BPMN, o que nem todas as ferramentas têm, é gratuita para utilização, e mais simples que a média das ferramentas.

A ferramenta necessita de uma instalação Java para operar normalmente. Ela foi testada e homologada na versão de Java 5. Pode-se baixar um JRE, que é menor requer menor esforço na instalação através do link [http://java.sun.com/javase/downloads/index\\_jdk5.jsp](http://java.sun.com/javase/downloads/index_jdk5.jsp).

Ela não foi testada com versões posteriores de Java, e não irá funcionar com versões anteriores. Uma vez baixado o executável, pode-se instalar como qualquer aplicativo Windows normal. Uma instalação típica pode ser selecionada, e será apresentada uma tela como :



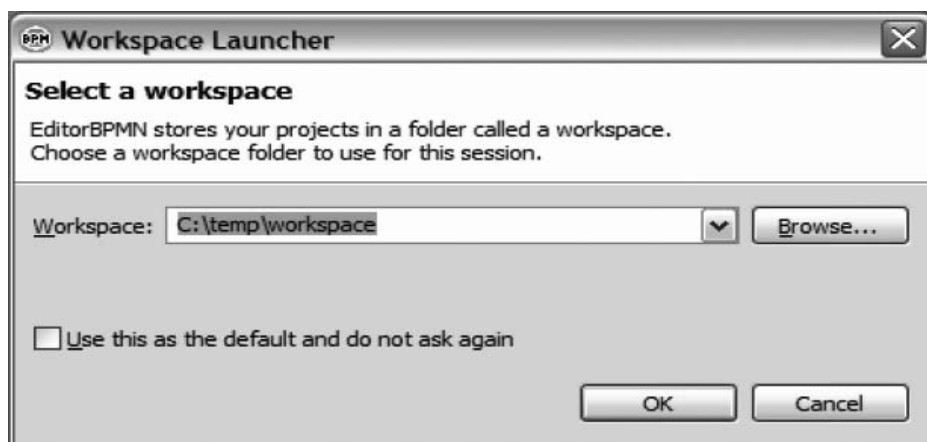
Após a instalação, você terá uma máquina virtual Java que será utilizada pelo programa.

A segunda etapa é baixar o programa de edição BPMN. Ele é fornecido como um ZIP, que pode ser aberto para qualquer diretório do computador. Deve-se criar um diretório na máquina, por exemplo c:\EditorBPMN.

Copia-se o arquivo ZIP para dentro desta pasta, e se extrai o conteúdo para dentro deste diretório. Após este processo, o diretório deve parecer como :

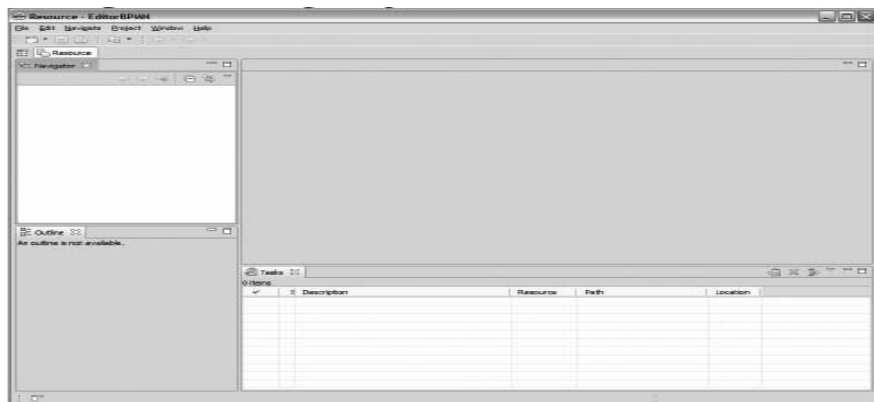
Nome	Mo
configuration	15/0
plugins	15/0
.eclipseproduct	15/0
EditorBPMN.exe	15/0
editorbpmn.zip	16/0
startup.jar	15/0

O arquivo editorbpmn.zip pode ser removido sem problemas agora. A instalação está completa. O executável EditorBPMN.exe irá iniciar o programa. Ele pode ser arrastado para o desktop para que seja criado um atalho, facilitando execuções futuras. Quando se executa o programa, será solicitado um diretório de trabalho, onde todos os arquivos criados serão armazenados :

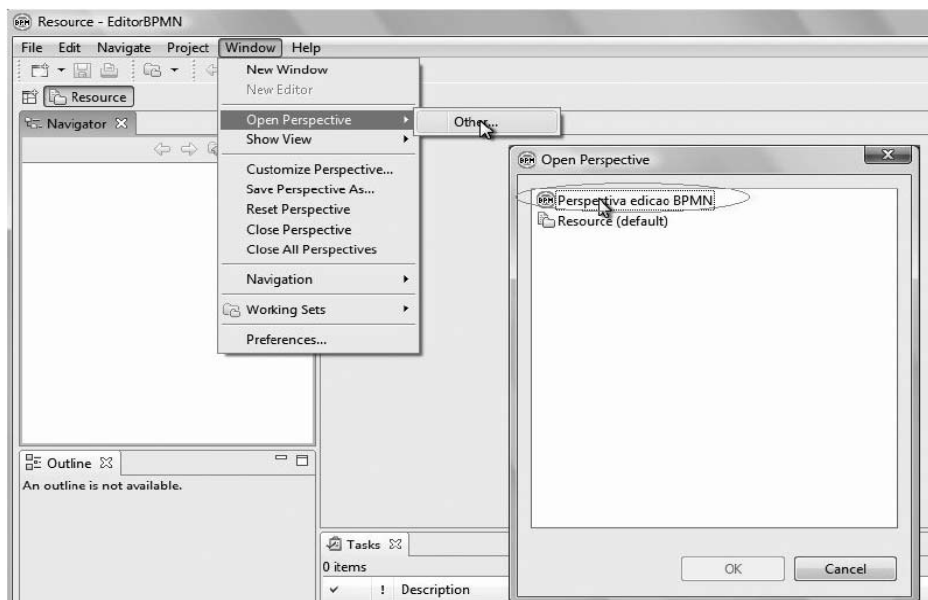




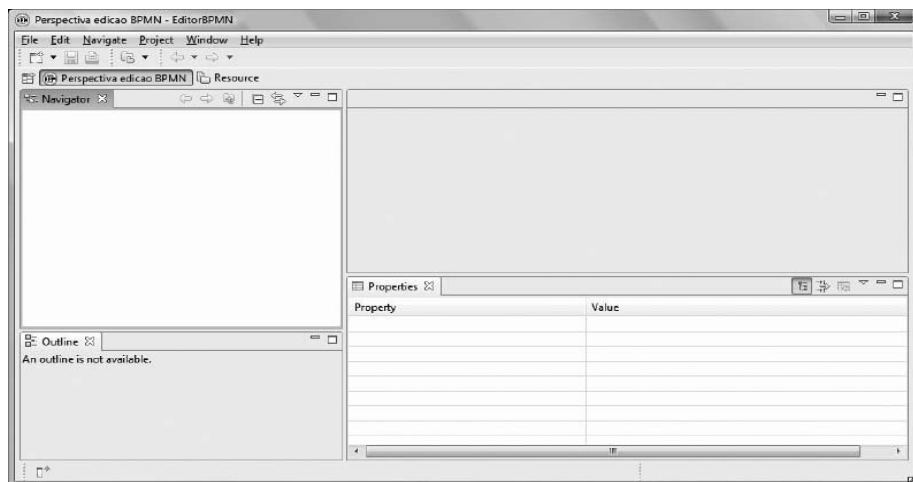
Após definido um espaço de trabalho, o programa entra em sua tela principal. A ferramenta foi construída com base no eclipse, e é formada por várias telas que interagem com a aplicação.



Para que tenhamos um ambiente customizado para utilização, podemos chavear para a perspectiva de edição BPMN. Isto pode ser feito selecionando os menus Window->Open perspective->Other, e seleciona-se "Perspectiva edição BPMN", como a seguir :

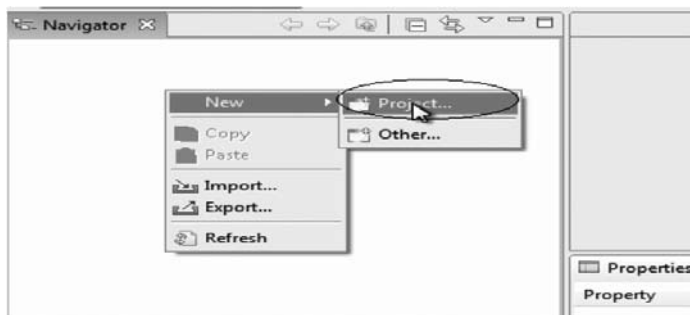


Após este chaveamento, a aparência ficará como :

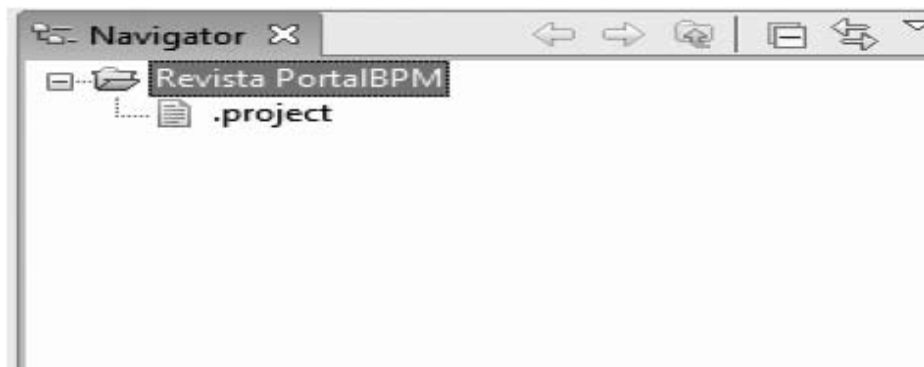


A ferramenta está pronta para uso. Nas entradas seguintes, a perspectiva não precisará mais ser chaveada. Se por alguma razão alguma janela for fechada ou a perspectiva perdida, pode-se novamente abrir a perspectiva com o procedimento acima.

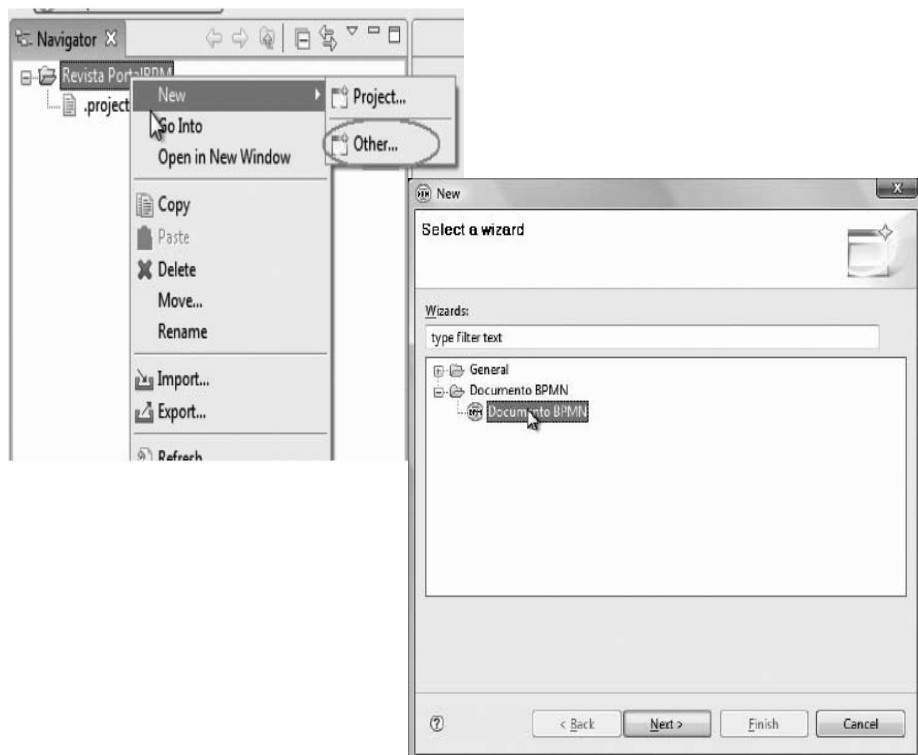
Todos os desenhos BPMN ficam dentro de um projeto. Precisamos criar ao menos um projeto, mas por outro lado podemos ter vários projetos em um workspace. A ferramenta pode também ter vários workspaces, bastando na entrada se selecionar diretórios diferentes para cada execução. A criação de um projeto é feita clicando-se com o botão da direita na janela Navigator, que está no canto superior direito da tela :



Será solicitado um nome para o projeto, e após criado irá apresentar uma aba em navegador, semelhante a uma estrutura de diretórios do Windows Explorer.

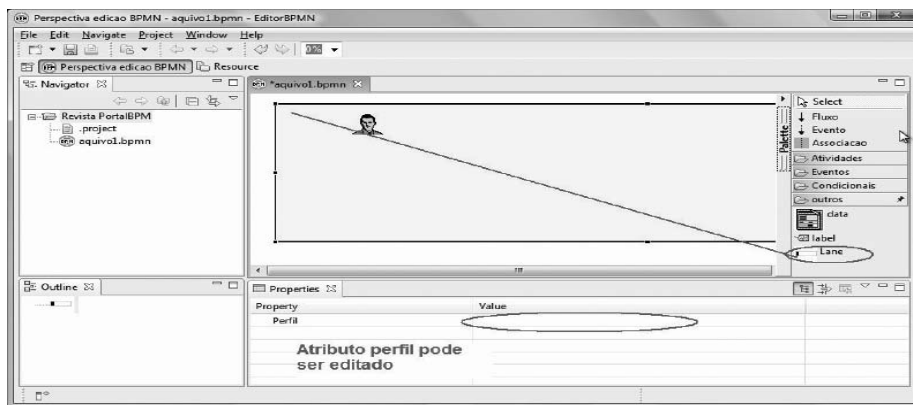


Dentro deste projeto podem ser criados vários desenhos BPMN. Selecionando-se o projeto com o botão direito do mouse será apresentado :



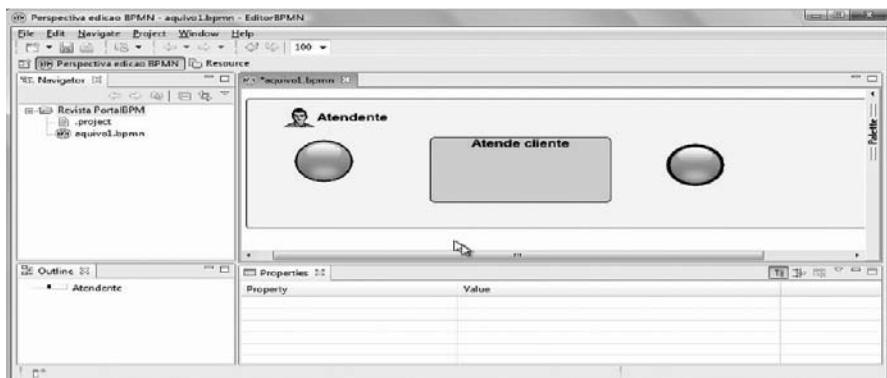


Vamos editar um exemplo simples de BPMN. A partir do editor aberto, selecionamos uma lane, que está na paleta em outros. Quando se clica em um sub-elemento da paleta, ele é aberto e fechado a cada interação. Leva-se a lane para a área de edição, e coloca-se na tela a tela ficará como :



Pode-se selecionar o valor de perfil e colocar um valor significativo para a Lane, como Atendente. Imediatamente o desenho na edição superior reflete este valor, bem como a janela de Outline.

O próximo passo é fazer o mesmo com os eventos de início e final, que devem ser arrastados sobre a Lane. Deve-se fazer o mesmo com uma atividade simples, que terá seu atributo texto alterado para "Atende cliente". Após estes passos, a tela deverá estar como :



A paleta pode ser aberta ou fechada, bastando-se clicar nela, ou posicinando-se o mouse por algum tempo sobre ela. Ela também se fecha automaticamente quando não está sendo utilizada.

Elementos BPMN podem ser arrastados para uma Lane, neste caso fazendo parte dela, ou seja, quando a Lane é arrastada todos os elementos internos são arrastados em conjunto. Pode-se também arrastar os elementos para uma área fora de uma lane, neste caso não pertencendo a nenhuma lane específica, funcionando como um processo privado, como explicado na primeira edição da revista PortalBPM.

Na parte superior da janela pode ser encontrado um selecionador de ZOOM, permitindo aumentar ou diminuir a imagem, facilitando a tarefa de visualização de elementos na tela.

A ferramenta permite exportar em formato imagem. Selecionando-se o mouse com o botão direito sobre o editor, em uma área não populada (branca), será apresentada uma opção “exporta diagrama como imagem”.

Pode-se apontar o nome do arquivo e será gerada uma imagem em JPG, que pode ser anexada a outros documentos textuais, como de requisitos.

Arquivos BPMN podem ser exportados da ferramenta, ou importados para ela.

Clicando-se com o botão da direita sobre um arquivo ou projeto, serão apresentados menus para exportação e importação. Isto permite compartilhar desenhos BPMN entre usuários.

Perceba que a cada alteração no diagrama, um asterisco aparece na lateral do nome do arquivo. Isto indica que o mesmo está alterado, e para salva-lo se clica em salvar no menu (o disquete na toolbar) ou o conjunto de teclas CONTROL+S. Neste momento o arquivo está salvo.

## **16 - Outras ferramentas de mercado**

Temos várias alternativas de mercado para desenho BPMN. Elas se dividem em ferramentas de desenho BPMN com BPMS integrado, e soluções puras de desenho.

Ferramentas de desenho com suporte ao BPMN

- **Toguether – Borland**
- **BEA – Aqualogic**
- **Intalio**
- **System Architect**
- **MetaStorm – Provisio**
- **Visual Paradigm**
- **WBI Modeler**
- **Visio**
- **Aris**

Atualmente temos cadastradas 44 ferramentas com suporte ao BPMN, e o número cresce a cada dia.

Não podemos deixar de citar o movimento Eclipse, que está construindo uma ferramenta de desenho BPMN sobre o GMF e GEF.

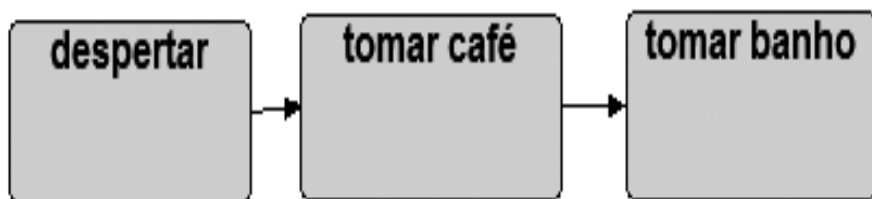
## 17 - BPMN design patterns

Da mesma forma que na orientação para objetos, os fluxos de processos desenhados em BPMN demandam padrões de desenvolvimento (design patterns).

Um design pattern, segundo a definição, é um trecho que aparece repetidas vezes durante os ciclos de desenvolvimento. Para facilitar a compreensão e garantir uma mesma linguagem durante as reuniões de levantamento, é importante utilizarmos os mesmos termos, evitando desta forma ambigüidades.

### Seqüência de fluxo (Sequence)

Este tipo de design pattern indica uma seqüência de fluxo. O mais óbvio é a seqüência, onde atividades são executadas em uma ordem, uma após a outra. A representação deste tipo de fluxo é algo como a seguir :



Neste caso, deve estar implícito que existe um token que vai sendo propagado conforme as atividades vão sendo executadas, sejam elas por um mesmo perfil ou por vários perfis (lanes) dentro da execução.

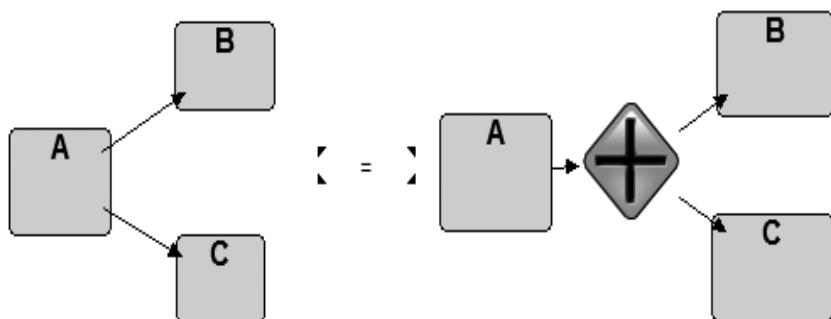


## Processamento paralelo (Parallel Split)

Quando seguimos por mais de um caminho ao longo da execução do fluxo, ou se temos atividades com tokens distintos (notação BPMN), chamamos de processamento paralelo (parallel split).

Isto indica que os dois caminhos são processados de forma independente. Em BPMN, temos mais de uma forma de representação para processamentos paralelos.

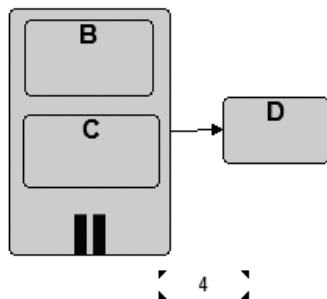
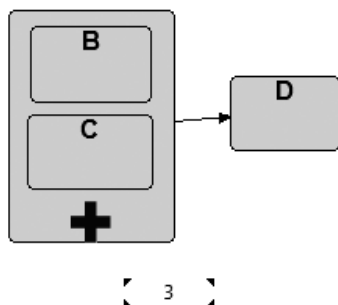
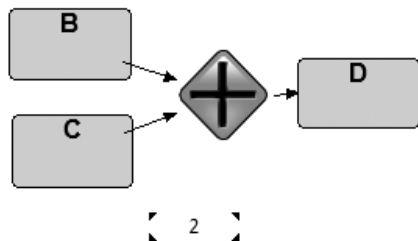
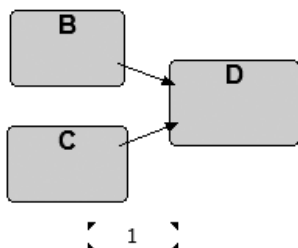
No exemplo a seguir, em todos os casos as atividades A e B irão gerar tokens distintos.



No exemplo à esquerda, temos um fluxo paralelo não controlado. Quando duas setas saem de uma atividade, isto indica processamento paralelo. No exemplo da direita, colocamos explicitamente o gateway de AND.

## Sincronização (Synchronization)

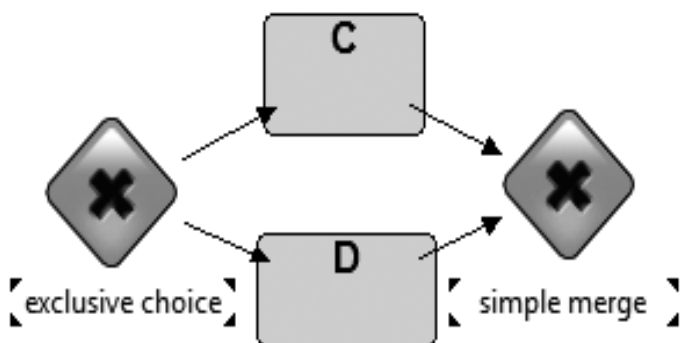
Utilizamos uma sincronização quando desejamos aguardar pela chegada de mais de um token gerado durante o processamento. Ele é a contrapartida do processamento paralelo. Suas representações podem ser :



A notação BPMN chama este tipo de padrão de Merge. Quando colocamos uma sincronização desejamos aguardar por um conjunto de processamentos antes de continuarmos.

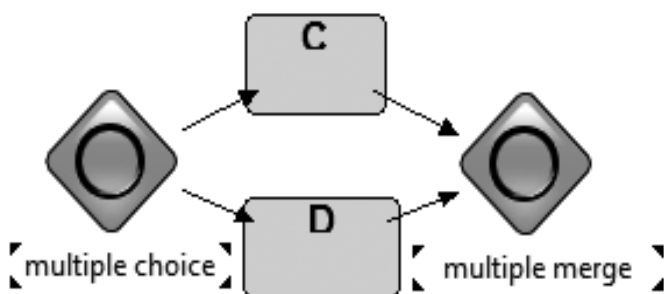
Escolha exclusiva (Exclusive Choice) e União simples (Simple merge)

Quando temos uma única escolha para um dos caminhos de um fluxo, através de um gateway XOR, definimos este padrão como Exclusive Choice. Sua contrapartida é o simple merge, que irá aguardar pela chegada do único token, e assim continuar o processamento. Este tipo de pattern indica que apenas um token foi gerado durante o processo.



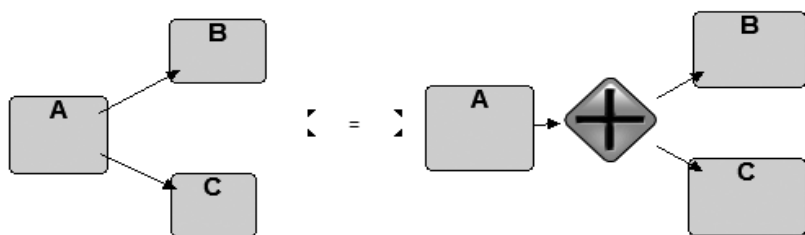
Múltipla escolha (Multiple choice) e Múltipla junção (Multiple merge)

Quando temos um mais caminhos sendo processados em um fluxo, indicamos através do pattern Multiple choice e Multiple merge.

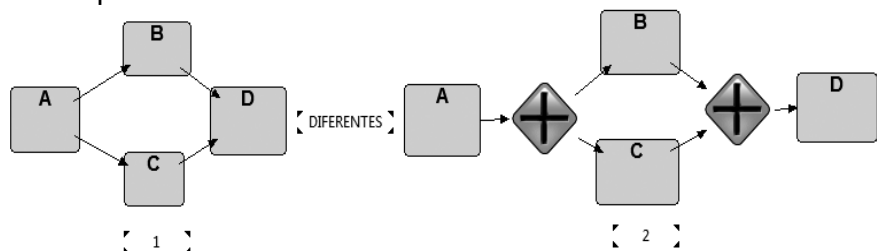


## Sobre junções e bifurcações

Os mecanismos de junção e bifurcação são completamente distintos, e não necessitam ser utilizados aos pares. Vamos compreender estes mecanismos através de alguns exemplos. Quando o diagrama contém um gateway para bifurcação ou junção, dizemos que este ambiente é controlado. Quando existe mais de um fluxo saindo ou chegando a uma atividade, dizemos que este ambiente é não controlado. Se mais de uma seta de fluxo sai de uma atividade, todos os caminhos são seguidos, como se fosse um AND.

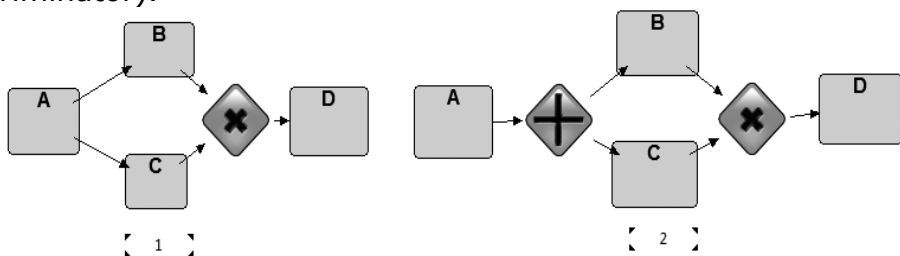


No exemplo anterior as duas representações são idênticas em termos de processamento. Dois tokens foram criados, seguindo para as atividades B e C. Já em termos de junção, um ambiente descontrolado gera resultados distintos, em termos de processamento.



Neste exemplo, tanto a implementação 1 como a 2 irão passar por A, gerar dois tokens que seguirão por B e C. A diferença é que no primeiro exemplo, cada um dos tokens que chegarem a D irão forçar a execução desta atividade. Portanto, no exemplo 1 D SERÁ EXECUTADO DUAS VEZES. Já no exemplo 2, como foi colocado o gateway para controle do ambiente, D será executado apenas uma vez, pois o gateway de junção irá aguardar a chegada dos dois tokens.

Mas desejamos às vezes que qualquer um dos tokens que chegue até a junção faça com que o fluxo continue. Este pattern é comumente chamado de Discriminação (discriminator).

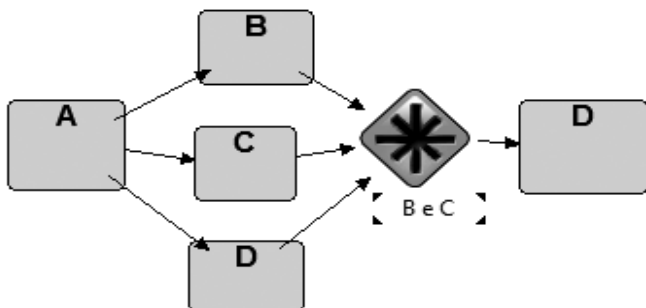


Neste caso, embora dois tokens tenham sido gerados, como a junção é um XOR, o primeiro dos tokens que chegar ao gateway fará com que o fluxo continue. O outro token, quando chegar ao gateway será descartado em termos de processamento.

Desta forma, é perfeitamente possível que a seqüência executada para o exemplo anterior seja A, B, D e C, isto mesmo, com a atividade D terminando antes mesmo da C.

### **Junção N de M (N out of M join)**

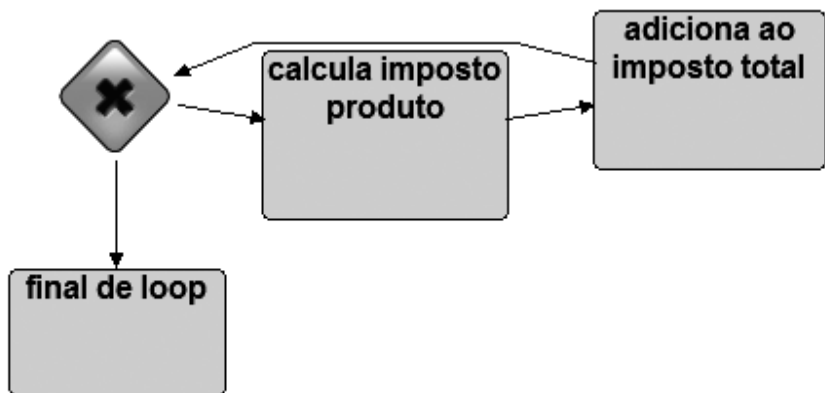
Mas, se a sincronização aguarda por todos os tokens, e a discriminação pelo primeiro que chegar, como podemos desenhar um fluxo de forma que alguns cheguem até o gateway para continuar ? Seria o “meio termo” entre a sincronização e discriminação.



Neste caso, desejamos que o fluxo continue quando as atividades B e C terminarem, e para isto utilizamos o gateway complex como junção. Na bifurcação ele funciona como forma de gerarmos várias saídas, e na junção como forma de escolha para quais elementos devem ser recebidos para que o fluxo continue.

### **Ciclo arbitrário (Arbitrary cycle)**

Quando desejamos executar um fluxo um certo número de vezes, mas não temos como precisar quantas vezes ele será executado, utilizamos um ciclo arbitrário. Ele funciona como uma atividade em paralelo, mas sem que definamos quantas vezes ele será executado.



É importante termos consciência destes patterns, mas acima de tudo conhecer os mecanismos de execução, para que não desenhemos um fluxo que no momento da execução funcione de uma forma diferente da esperada, apenas porque selecionamos um gateway de forma errônea.

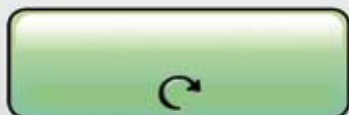
## Atividades



Atividade



Compensação



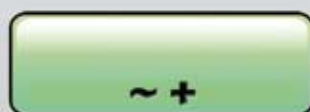
Loop



Paralelo



Sub atividade



Add Hoc

## Pools e Lanes



## Artefatos



Dados

A notação BPMN é a simbologia mais respeitada na atualidade para desenho de processos de negócios, e será a base das soluções BPMS do futuro.

Compreenda seu poder e como podemos expressar fluxos de forma muito mais clara.

Conheça os design patterns aplicados em fluxos de processos e os erros comuns no desenho com a notação, através da única literatura brasileira focada no padrão do futuro.