UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

Relatórios Técnicos
do Departamento de Informática Aplicada
da UNIRIO
n° 0020/2009

Estudos do Módulo Aris for SOA

Jairo Francisco de Souza
Leonardo Azevedo
Camille Furtado
Fernanda Baião
Flávia Santoro

Departamento de Informática Aplicada

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
Av. Pasteur, 458, Urca - CEP 22290-240
RIO DE JANEIRO – BRASIL

Projeto de Pesquisa

Grupo de Pesquisa Participante

Patrocínio

PETROBRAS

Relatórios Técnicos do DIA/UNIRIO, No. 0020/2009
Editor: Prof. Sean W. M. Siqueira Outubro, 2009

Estudos do Módulo Aris for SOA *

Jairo Francisco de Souza[1,3], Leonardo Azevedo[1,2], Camille Furtado[1], Fernanda Baião[1,2], Flávia Santoro[1,2]

[1]Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
[2]Departamento de Informática Aplicada (DIA) – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
[3]Departamento de Ciência da Computação (DCC) – Universidade Federal de Juiz de Fora (UFJF)

jairo.souza@ufjf.edu.br, azevedo@uniriotec.br, camillefurtado@gmail.com, fernanda.baiao@uniriotec.br, flavia.santoro@uniriotec.br

Abstract. Service Oriented Architecture is presented as being more flexible and capable to support services independent of platform and protocol in a distributed environment. In this work, we describe the SOA module of the ARIS tool, showing its functionalities and the aspects of SOA that the tool provides

Keywords: Service Oriented Architecture, service.

Resumo. A arquitetura orientada a serviços (SOA – Service Oriented Architecture) apresenta-se como sendo mais flexível e capaz de suportar serviços independentes de plataforma e protocolo em um ambiente distribuído. Neste trabalho, apresentamos o módulo SOA da ferramenta ARIS, explicitando suas funcionalidades e os aspectos de SOA que a ferramenta atende.

Palavras-chave: Arquitetura orientada a serviço, serviço.

* Trabalho patrocinado pela Petrobras.

Sumário

1 INTRODUÇÃO 1

1.1 FUNCIONALIDADES EXISTENTES NO MÓDULO SOA 1

1.2 ESTRUTURA DO RELATÓRIO 1

2 RECURSOS DO MÓDULO DE SOA DO ARIS 1

2.1 EXPORTAÇÃO DE DIAGRAMAS DE PROCESSOS BPEL EM ARQUIVOS BPEL 3

2.2 TRANSFORMAÇÃO DE DIAGRAMAS EPC PARA DIAGRAMAS DE PROCESSOS BPEL 3

2.3 IMPORTAÇÃO DE SERVIÇOS (WSDL) 4

2.4 VALIDAÇÃO DE DIAGRAMAS EPC 6

2.4.1 Regras estruturais 6

2.4.2 Regras para EPC orientados a serviços 6

2.5 MODELAGEM DE XML SCHEMA DEFINITIONS (XSD) 7

2.6 IMPORTAÇÃO DE ESTRUTURA DE DADOS 9

3 EXEMPLO DA CRIAÇÃO DE UM MODELO BPEL A PARTIR DE UM EPC ... 11

3.1 EXEMPLO 1: ALTERANDO MINIMAMENTE O DIAGRAMA EPC 14

3.2 EXEMPLO 2: UTILIZANDO O MÁXIMO DE INFORMAÇÃO DO DIAGRAMA EPC 23

4 AVALIAÇÃO DO USO DA FERRAMENTA NO PROJETO 31

GLOSSÁRIO 32

REFERÊNCIAS 33

APÊNDICE A - NOTAÇÃO BPEL UTILIZADA NO MÓDULO ARIS 34

1 Introdução

Este documento apresenta o módulo SOA da ferramenta ARIS, explicitando suas funcionalidades e os aspectos de SOA que a ferramenta atende. O documento foi baseado no manual do ARIS SOA Designer 7.0 [ARIS, 2006] e foi produzido pelo Projeto de Pesquisa em SOA como parte das iniciativas dentro do contexto do Projeto de Pesquisa do Termo de Cooperação entre NP2Tec/UNIRIO e a Petrobras/TIC-E&P/GDIEP.

Dado que o módulo SOA do ARIS fornece uma notação gráfica para os padrões WSDL versão 1.1 [WSDL, 2001] e BPEL4WS versão 1.1 [BPEL4WS, 2003], o conhecimento destes dois padrões é fundamental para compreender os detalhes da notação gráfica do ARIS.

1.1 Funcionalidades existentes no módulo SOA

O módulo SOA da ferramenta ARIS possui as seguintes funcionalidades:

- Exportação de diagramas de processos BPEL em arquivos BPEL
- Transformação de diagramas EPC para diagramas de processos BPEL
- Importação de serviços (WSDL)
- Validação de diagramas EPC
- Modelagem de XML Schema Definitions (XSD)
- Importação de estrutura de dados

Essas funcionalidades serão descritas no capítulo 2.

1.2 Estrutura do Relatório

Esse relatório está organizado em 6 capítulos, sendo o capítulo 1 a Introdução.

No capítulo 2 são apresentados os recursos do módulo SOA do ARIS.

No capítulo 3 é apresentado um exemplo prático de utilização do módulo SOA para transformação de um diagrama EPC em um diagrama de processo BPEL.

No capítulo 4 são apresentadas nossas conclusões sobre o módulo.

Por fim, os capítulos 5 e 6 apresentam o glossário e as referências bibliográficas, respectivamente.

2 Recursos do módulo de SOA do ARIS

O módulo SOA da ferramenta ARIS tem como principal utilidade a representação gráfica dos elementos BPEL[1] na versão 1.1. Essa representação gráfica permite o

[1] A especificação 1.1 do BPEL é chamada também de BPEL4WS (BPEL4WS, 2003), enquanto a

entendimento do processo BPEL de forma mais clara e evita erros de digitação e/ou sintaxe na construção de arquivos BPEL.

A definição BPEL 1.1 não especifica notações gráficas. Assim, o módulo SOA utiliza notação gráfica criada pela própria IDS Sheer, a qual é apresentada no Anexo I.

O módulo SOA fornece dois diagramas para representação BPEL:

- Diagrama de processo BPEL: usado para representar um processo. A Figura 1 apresenta um exemplo de diagrama de processo BPEL, onde podem ser visualizadas as atividades do negócio a serem transformadas em tividades no BPEL e sua sequência de execução. Os detalhes de cada atividade estão no diagrama de alocação BPEL, descrito a seguir (Figura 2).
- Diagrama de alocação BPEL: usado para detalhar uma atividade dentro de um processo BPEL. Na Figura 2 é apresentado um exemplo de diagrama de alocação ligado à atividade InitiatePriceCalculation. As propriedades da atividade estão definidas neste diagrama.

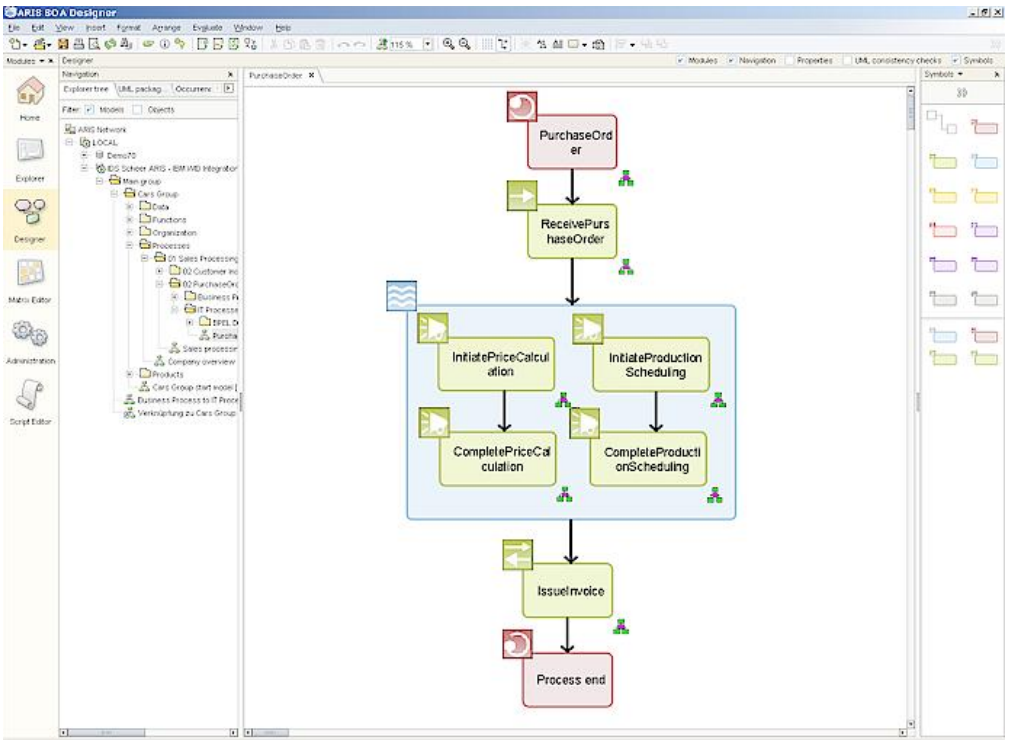


Figura 1 – Diagrama de processo BPEL

especificação 2.0 é chamada de WS-BPEL (WS-BPEL, 2007). O termo BPEL é utilizado como parte do nome de diagramas no ARIS, indiferente aos dois nomes possíveis da especificação. Neste documento, utilizaremos o termo BPEL para designar elementos referentes ao BPEL4WS.

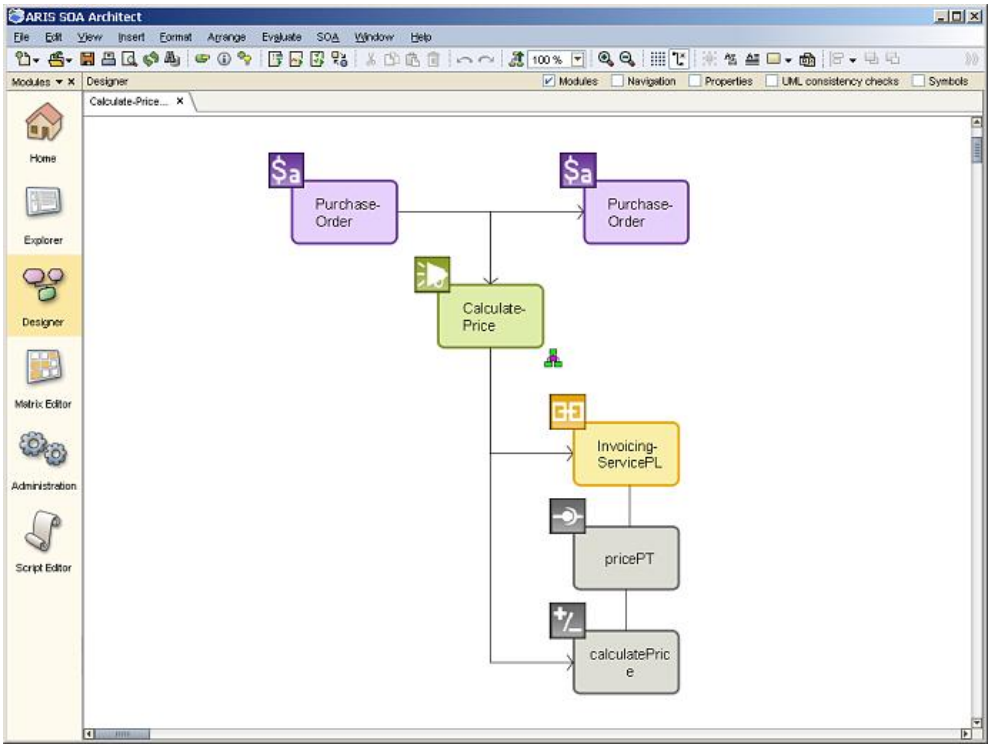


Figura 2 – Diagrama de alocação BPEL

2.1 Exportação de diagramas de processos BPEL em arquivos BPEL

O módulo SOA permite exportar os diagramas de processo BPEL para disponibilizá-los para outros sistemas, ou seja, os arquivos gerados pela exportação podem ser lidos por um barramento de serviço (como BEA, Oracle SOA Suíte, Microsoft, etc) que os interpretará e controlará a execução do processo conforme foi especificado no diagrama. Dois arquivos com o mesmo nome do modelo são exportados – um deles com a extensão **bpel** e o outro com a extensão **wsdl**. Se forem usados *web services* em um dos diagramas BPEL para os quais não for definido um PartnerLinkType (vide Figura 81), então os PartnerLinkType são gerados como arquivos de empacotamento com o nome do respectivo *web service* e extensão **wsdl**. Todos os arquivos são empacotados automaticamente em arquivos Zip. No ARIS SOA Designer, somente é possível exportar os modelos do tipo diagrama de processo BPEL para os formatos BPEL/WSDL.

2.2 Transformação de diagramas EPC para diagramas de processos BPEL

O módulo ARIS permite gerar um diagrama de processos BPEL utilizando como fonte um diagrama EPC (*Event-driven Processo Chain*). Durante a transformação de um diagrama EPC para um modelo de processo BPEL, os seguintes elementos são gerados:

- um novo diagrama de processo BPEL e seus diagramas de alocação, um para cada atividade;
- uma atividade de invocação para cada atividade automatizada2 do EPC;
- uma nova representação dos serviços do processo BPEL nos seguintes diagramas: UML Component Diagram (Figura 4) e Access Diagram (Figura 3);
- uma atividade BPEL associada a um diagrama de processo BPEL, caso o EPC contenha uma atividade associada a este diagrama de processo BPEL.

2.3 Importação de serviços (WSDL)

O módulo SOA permite importar definições de serviços, descritos em linguagem WSDL. Para acessar essa definição é preciso fornecer uma URI (ou seja, pode-se fornecer uma URL para a definição do serviço, utilizar um arquivo .wsdl que esteja gravado em disco etc).

Ao importar uma definição de serviço, é possível gerar uma visão de negócio e/ou uma visão de TI desse serviço.

- A visão de negócio contém os aspectos funcionais do serviço (como mostra a Figura 3), e é representada através do modelo do ARIS denominado Access Diagram
- A visão de TI contém os aspectos técnicos do serviço, como mostra a Figura 4, e é representada através de um modelo do ARIS denominado UML Component Diagram.

As definições de serviço em WSDL podem importar ou conter XML Schema Definitions (XSD) (ver seção 2.5). Neste caso, os XSDs são importados e é criado um grupo (pasta) adicional para os dados do XSD do serviço.

A Figura 3 ilustra a inter-relação entre a visão funcional e a técnica do *web service* Scheduling em um **Access diagram** no ARIS. O componente de sistema **Scheduling** é conectado ao componente UML **Scheduling** por uma conexão do tipo **encompasses** (encapsula).

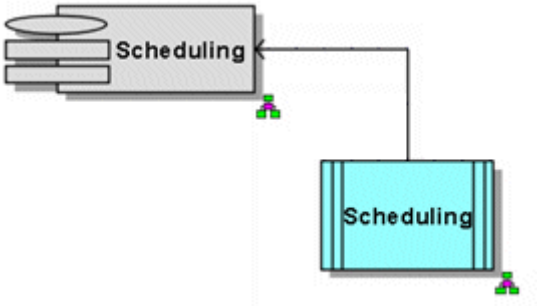


Figura 3 – Visão funcional do serviço importado Scheduling

A representação UML desse serviço é ilustrada na Figura 4 abaixo.

[2] Atividade automatizada é aquela executada computacionalmente.

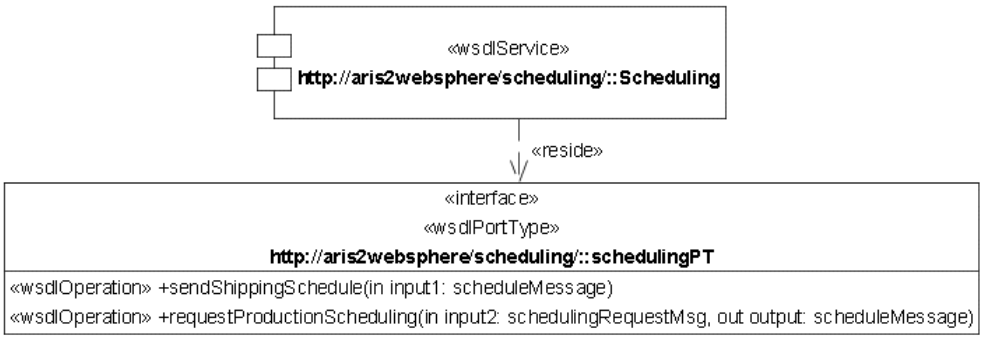


Figura 4 – Visão técnica do serviço importado Scheduling

Na Figura 4 temos o serviço Scheduling e a descrição de sua interface denominada schedulingPT que possui duas operações, sendShippingSchedule e requestProductionScheduling. A operação sendShippingSchedule recebe uma mensagem de entrada chamada input1 do tipo scheduleMessage e a operação requestProductionScheduling recebe uma mensagem de entrada do tipo schedulingRequestMsg e envia uma mensagem de saída do tipo scheduleMessage.

Depois da importação, é possível que alguns grupos sejam criados abaixo do grupo no qual o *web service* foi importado (Figura 5 e Figura 6). Os objetos do *web service*, tais como interface, componentes, classes e conexões, por exemplo, estão armazenados abaixo do pacote <nome do *web service*>/wsdl. Os vários tipos de dados usados no *web service* são criados abaixo da pasta <namespace>/xsd.

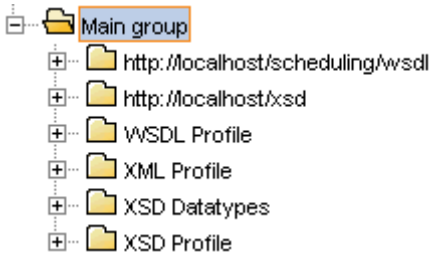


Figura 5 – Estrutura de grupos criados após a importação de um *web service (ARIS Explorer)*

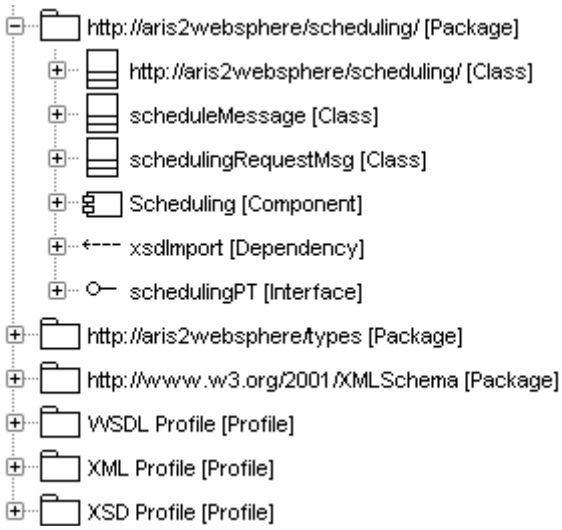


Figura 6 – Estrutura de grupos criados após a importação de um *web service (ARIS UML package)*

2.4 Validação de diagramas EPC

A validação de diagramas EPC é realizada para verificar se um diagrama EPC pode ser transformado em um diagrama de processo BPEL. Para isso, são verificadas algumas regras estruturais do EPC e regras específicas de um EPC orientado a serviço. Segue abaixo uma listagem das regras impostas pelo módulo ARIS.

2.4.1 Regras estruturais

- Todas as funções de negócio ou eventos devem possuir apenas uma conexão de entrada ou saída;
- Todo caminho deve começar e terminar com um evento;
- Nenhum caminho pode ser criado com operadores OR ou XOR após um evento;
- Não podem existir objetos sem conexões. Cada objeto no modelo deve possuir um ou mais objetos predecessores ou sucessores;
- Todo operador lógico deve ter uma e somente 1 conexão de entrada e um mínimo de 2 conexões de saída ou um mínimo de 2 conexões de entrada e somente 1 conexão de saída.

2.4.2 Regras para EPC orientados a serviços

- Uma função de sistema[3] é suportada por um único objeto do tipo "Application system type": uma função de sistema pode ser suportada por apenas um

[3] Uma função de sistema representa uma atividade automatizada.

único objeto do tipo "Application system type" e então ser interpretada na transformação para um processo BPEL;

- Todos os objetos de entrada e saída precisam ser mapeados para objetos do tipo "Class" ou serem objetos do tipo "Class": todos os objetos que são conectados com funções através de relacionamentos do tipo "has input" ou "has output" devem ser do tipo "Class" ou serem direta ou indiretamente mapeados para um ou mais objetos do tipo "Class" para serem interpretados corretamente na transformação de um processo BPEL;
- Um objeto do tipo "application system type" que suporta uma função de sistema pode ser conectado apenas com um objeto do tipo "component" (vide Figura 3). De acordo com as convenções da modelagem para representação de serviços no ARIS, apenas um único objeto do tipo "component" pode ser conectado a um objeto do tipo "application system type" para ser interpretado na transformação de um processo BPEL;
- Apenas um único objeto do tipo "operation" pode ser conectado a uma função de sistema e ser interpretado na transformação de um processo BPEL.
- Os processos contêm atividades de mensageria pública: atividades públicas de mensageria servem para especificar a informação de entrada e saída do processo. Esta informação é usada por outros componentes quando forem se comunicar com o processo. De acordo com as convenções da modelagem para EPC's orientados a serviço, atividades públicas de mensageria podem representar passos do processo que seguem eventos de início ou fim e especificam os objetos de dados de entrada ou saída associados do processo. Ainda, os dados de entrada e saída do processo BPEL podem ser especificados como clusters do evento inicial ou do evento final do EPC.
- Fluxos paralelos de processo, caminhos de decisão inclusivos e exclusivos devem ser bem-formados: os fluxos paralelos do processo devem ser especificados dividindo ou agrupando caminhos usando operadores AND ou XOR. Os fluxos paralelos conterão ou uma operação AND/XOR sendo aberta e que não conterá nenhuma outra conexão entre os dois caminhos criados; ou conterão uma operação AND/XOR sendo fechada e que será a junção de todas as conexões (caminhos).

2.5 Modelagem de XML Schema Definitions (XSD)

Os web services devem ser capazes de trocar dados e fazer chamadas a funções em computadores remotos. Para que isso ocorra, o formato dos dados deve ser definido e isso é feito usando um *XML Schema Definition* (XSD). O módulo SOA do ARIS permite modelar as definições de dados graficamente. A Figura 7 abaixo mostra um exemplo de um arquivo XSD.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://aris2websphere/types">
 <xsd:complexType name="customerInfo">
  <xsd:sequence>
   <xsd:element name="info" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="purchaseOrder">
  <xsd:sequence>
   <xsd:element name="order" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="invoice">
  <xsd:sequence>
   <xsd:element name="info" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="shippingInfo">
  <xsd:sequence>
   <xsd:element name="info" type="xsd:string"/>
  </xsd:sequence>
```

Figura 7 – XML Schema Definition - XSD

As definições de dados são criadas no diagrama de classes da UML (Class diagram). Um exemplo de diagrama de classe representando **XSD** é apresentado na Figura 8, para o caso do tipo complexo "PurchaseOrder".

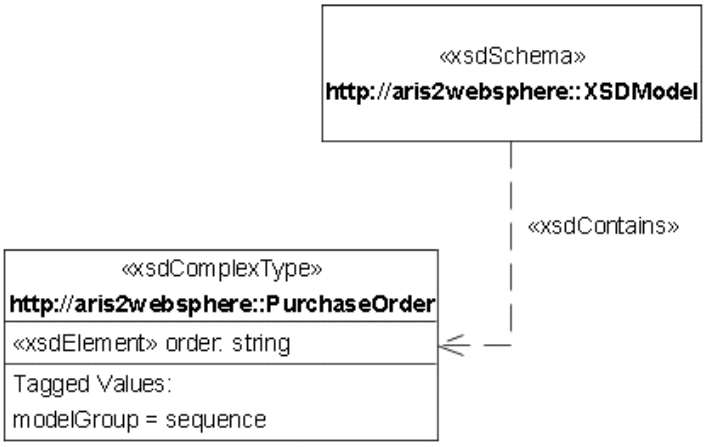


Figura 8 – Diagrama de classe representando XSD

2.6 Importação de estrutura de dados

O módulo ARIS permite importar estrutuda de dados descritos como XML Schema Definitions (XSD). Para tal, é preciso estar logado em um banco de dados que possua o filtro **ARIS for SOA** e a importação do arquivo "SOA profiles.xml" deve ter sido realizada.

O XSD é importado pro repositório do ARIS e um diagrama de classe UML é criado automaticamente durante sua importação para prover uma representação gráfica do XSD, como mostra a Figura 9.

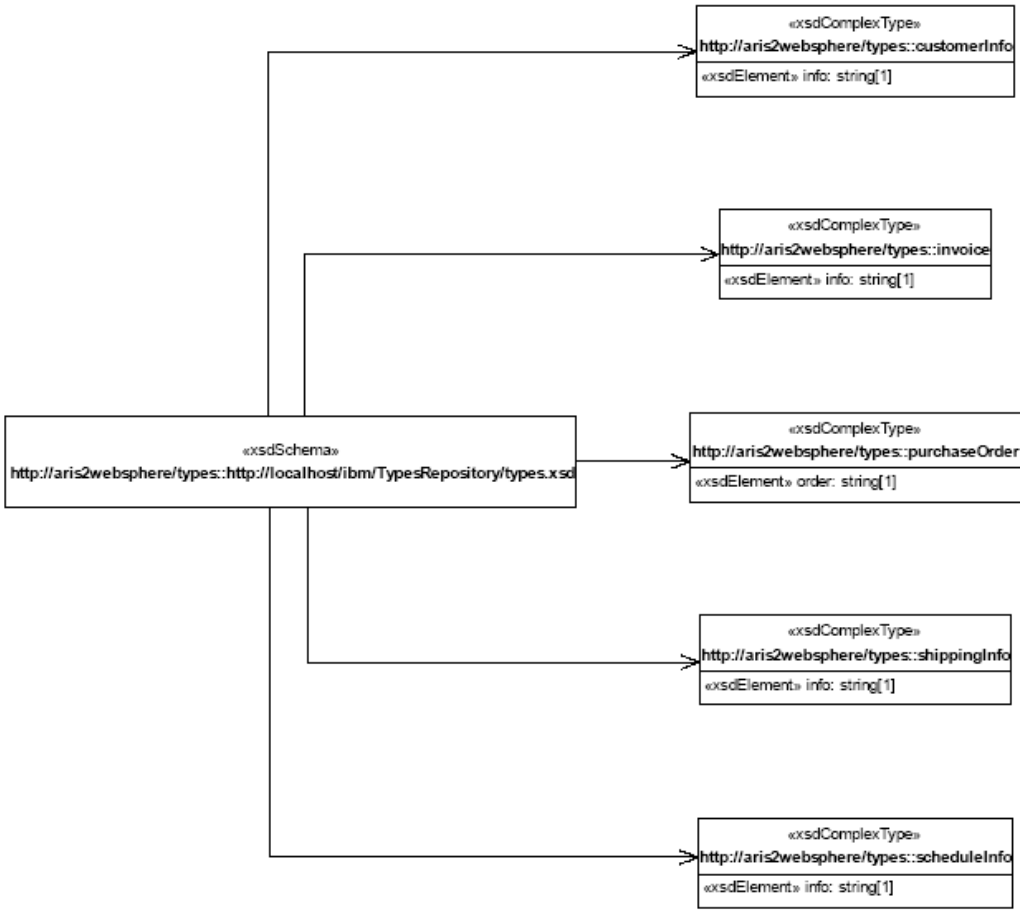


Figura 9 – XSD como um diagrama de classe UML no ARIS

A importação de um XSD atualiza os dados em um determinado *namespace* ou cria um grupo abaixo do grupo selecionado para importação. As seguintes regras se aplicam:

- Se o esquema tiver um namespace alvo4, o subgrupo recebe esse nome. Se ele não tem um namespace alvo, o subgrupo recebe o nome do arquivo XSD.
- Se um subgrupo com o mesmo nome já existir, ele é reutilizado. Se não existir, é criado.

[4] Corresponde ao atributo targetNamespace da linguagem XML, que indica qual o espaço de nomes (*namespace*) principal utilizado no documento XML.

Após a importação, o subgrupo conterá todos os dados importados. Se um subgrupo reutilizado já contiver elementos UML que representem os dados sendo importados, eles são atualizados de acordo com os seguintes critérios:

- Se não existir nenhum elemento UML para um dado elemento no schema, ele é criado.
- Se o elemento UML já existir e tiver no XSD, as propriedades do elemento são atualizadas.
- Se o elemento UML existe, mas não existem ainda elementos do XSD, o elemento XML é apagado do repositório UML (mas não do repositório ARIS). O elemento UML apenas é apagado se tiver sido modelado como parte da definição do esquema XSD que está sendo definido.
- A verificação se um dado elemento já existe é feita pelo atributo qualified name.

As seguintes regras se aplicam para a criação e atualização de elementos:

- Os elementos UML são criados de acordo com a convenção de modelagem do esquema UML.
- Se o XSD importado não possuir um namespace, um pacote com o estereótipo é criado com o nome xmlNoNamespace.
- Se um XSD importado possuir referências cruzadas a outro esquema de definições que ainda não tenha sido importado, esses documentos adicionais também serão importados. Se os elementos referenciados pelo outro XSD não existirem no banco, eles serão criados.

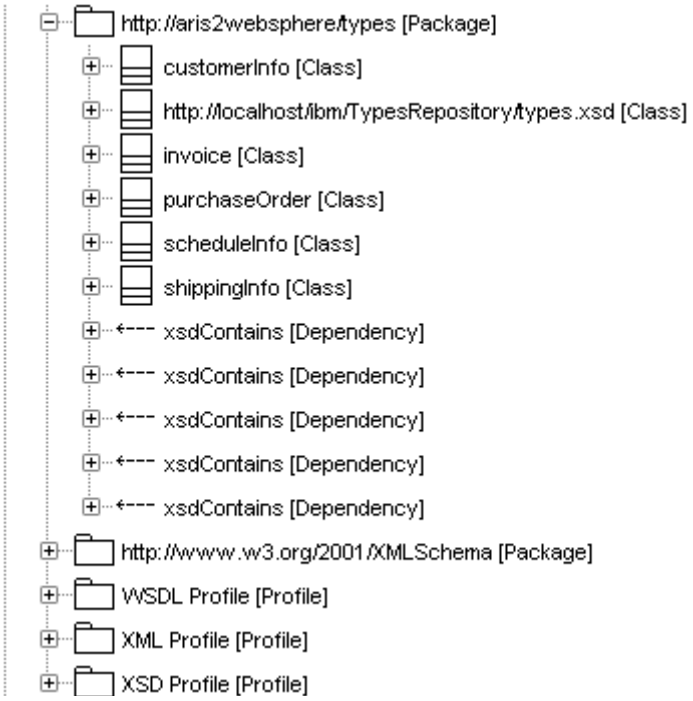


Figura 10 – Estrutura de grupos após a importação de um XSD

A Figura 10 mostra a estrutura de grupos na árvore de pacotes UML que é criada no ARIS após a importação do XSD representado graficamente na Figura 9.

3 Exemplo da criação de um modelo BPEL a partir de um EPC

Essa seção apresenta um exemplo de criação de um modelo BPEL utilizando como base um modelo de processos descrito em um EPC. Será utilizado como exemplo o diagrama de processo apresentado na Figura 11, o qual descreve um processo para compra. O processo de compra possui 6 atividades, onde cada atividade está ligada a um diagrama FAD. Abaixo, seguem os diagramas:

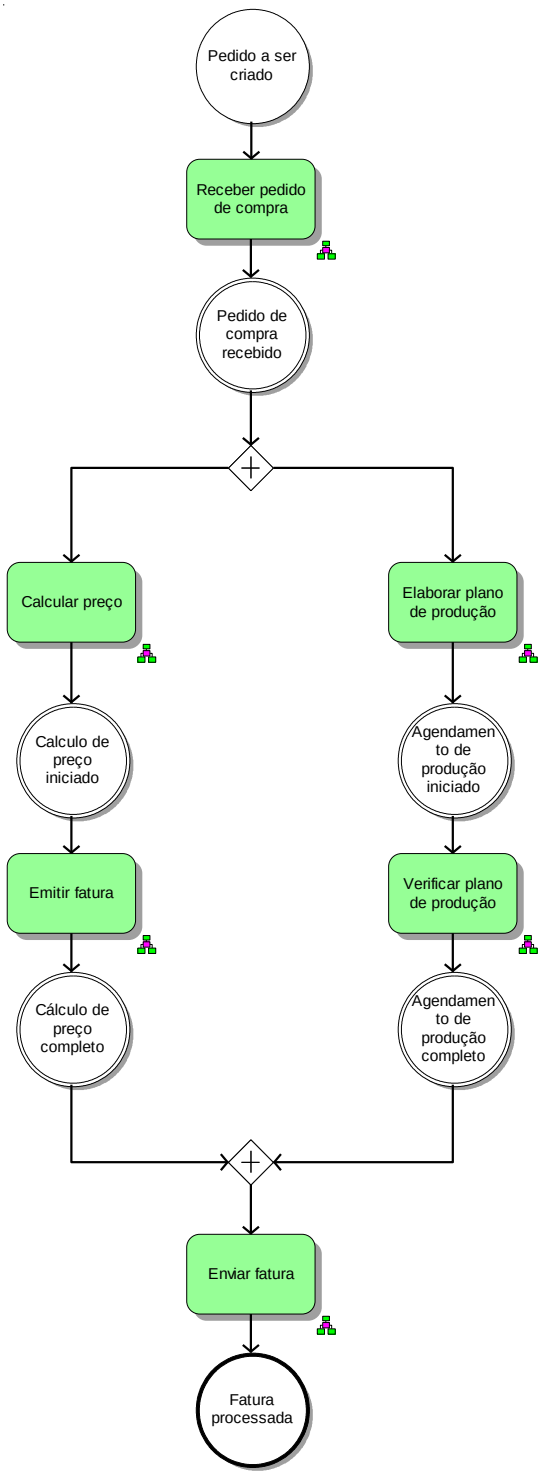


Figura 11 – EPC Comprar item

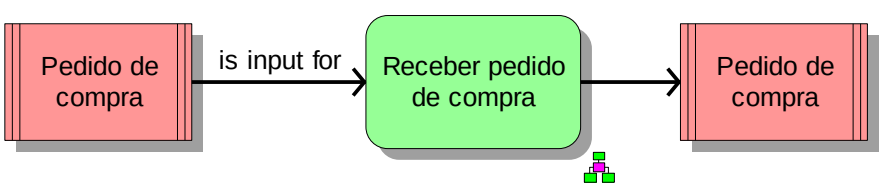


Figura 12 – FAD Receber pedido de compra

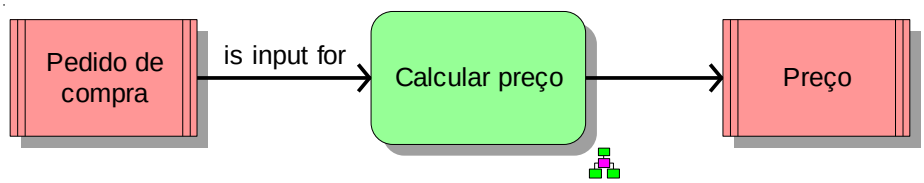


Figura 13 – FAD Calcular preço

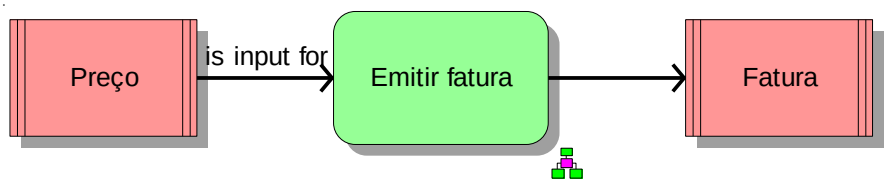


Figura 14 – FAD Emitir fatura

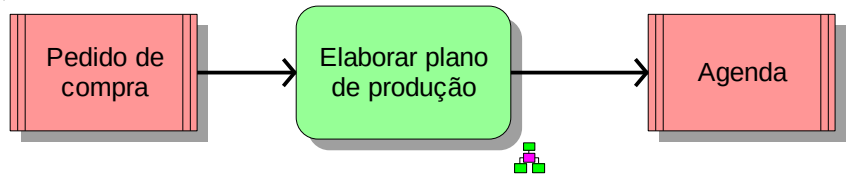


Figura 15 – FAD Elaborar plano de produção

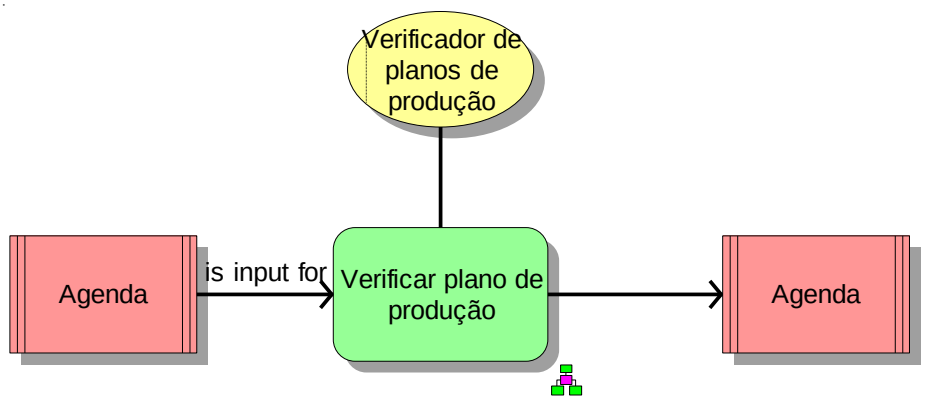


Figura 16 – FAD Verificar plano de produção

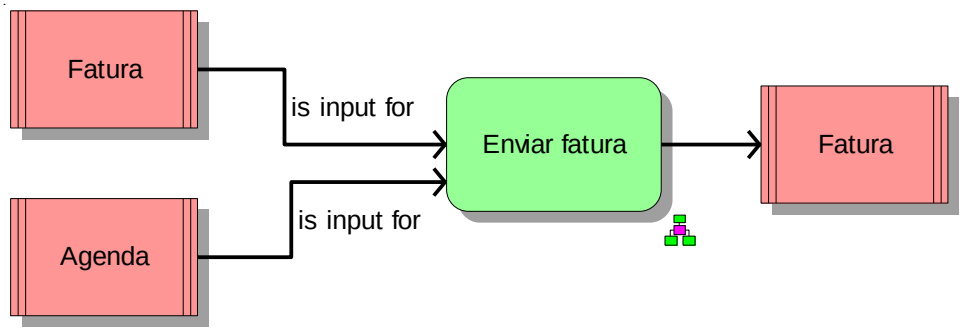


Figura 17 – FAD Enviar fatura

O EPC será transformado em um EPC orientado a serviço, isto é, um EPC onde cada uma da suas atividades é considerada uma chamada de serviço e, além disso, é modelado de tal forma que o módulo SOA possa transformá-lo em um diagrama BPEL. Contudo, um diagrama EPC não modelado corretamente para o formato esperado na transformação não produz erros durante a transformação para um diagrama BPEL e, sim, resulta em um diagrama com informações incompletas que pode, futuramente, ter suas informações completadas diretamente no diagrama de processos BPEL. Para elucidar as diferenças de modelagem, esse exemplo está dividido em duas seções. Na primeira seção, será exemplificada a transformação do diagrama EPC da Figura 11 alterando-o minimamente para que o impacto quanto às diretrizes de modelagem de processos na E&P [E&P, 2008] seja pouco afetado. Na segunda seção, será exemplificada a transformação do mesmo diagrama EPC, porém realizando as alterações necessárias para que o máximo de informação possa ser utilizada na sua transformação em um diagrama de processo BPEL e, assim, menos informação será necessário acrescentar manualmente no diagrama de processo BPEL.

3.1 Exemplo 1: alterando minimamente o diagrama EPC

Vamos considerar que todas as atividades desse processo poderão ser automatizadas. Porém, no momento, possuímos somente um serviço já criado e pronto para ser usado. Dessa forma, poderemos verificar como a ferramenta trabalha com atividades automatizadas na modelagem e atividades que ainda serão automatizadas. O único serviço disponível é BNPrice, o qual é responsável por calcular o preço do pedido. Esse serviço foi importado na ferramenta ARIS através da opção Import Service Description e possui somente uma porta (portType) chamada BNPriceSoap com a operação GetBNQuote disponível.

Como primeiro passo, iremos relacionar o serviço BNPrice à atividade CalcularPreço, ou seja, o cálculo do preço será feito pelo serviço BNPrice. Para tal, basta abrirmos o menu de contexto e selecionar a opção Select Service, a qual exibe o menu exposto na figura abaixo.

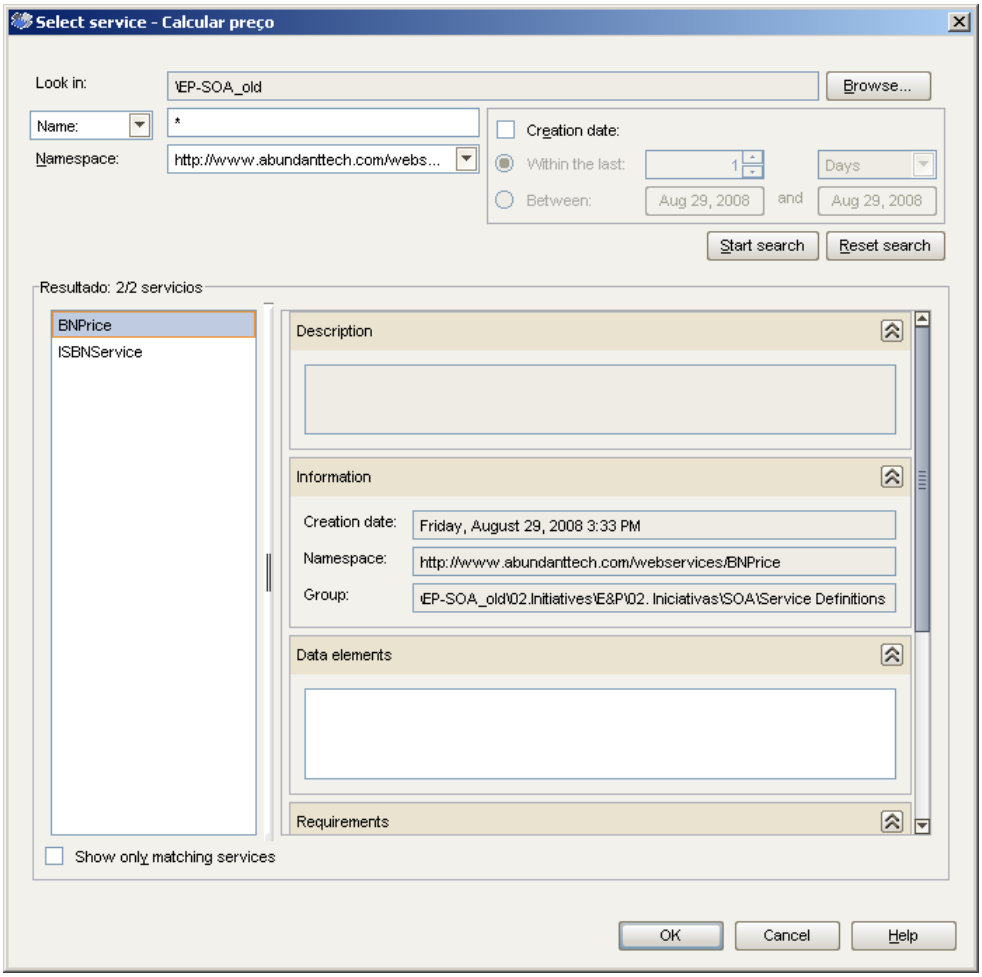


Figura 18 – Tela para seleção de serviços

A opção Select Service só está disponível no diagrama EPC e, por isso, não pode ser acessada dentro de um FAD, como o FAD correspondente da atividade Calcular Preço (Figura 13). Uma vez selecionado o serviço BNPrice, o EPC Comprar item é alterado, como mostra a Figura 19. A atividade Calcular Preço teve seu formato alterado, indicando que agora esta é uma atividade realizada por um sistema e o serviço BNPrice está ligada a essa atividade. Neste ponto, temos uma mudança entre as diretrizes de modelagem de processos atualmente em uso na E&P [E&P,2008] e a forma de modelagem de EPC orientado a serviço para o módulo SOA. Note, contudo, que a figura acima não contém informações sobre as operações que o serviço fornece. De fato, a associação de um serviço a uma atividade não informa qual operação apóia a atividade, somente qual serviço fornece a operação que a atividade necessita.

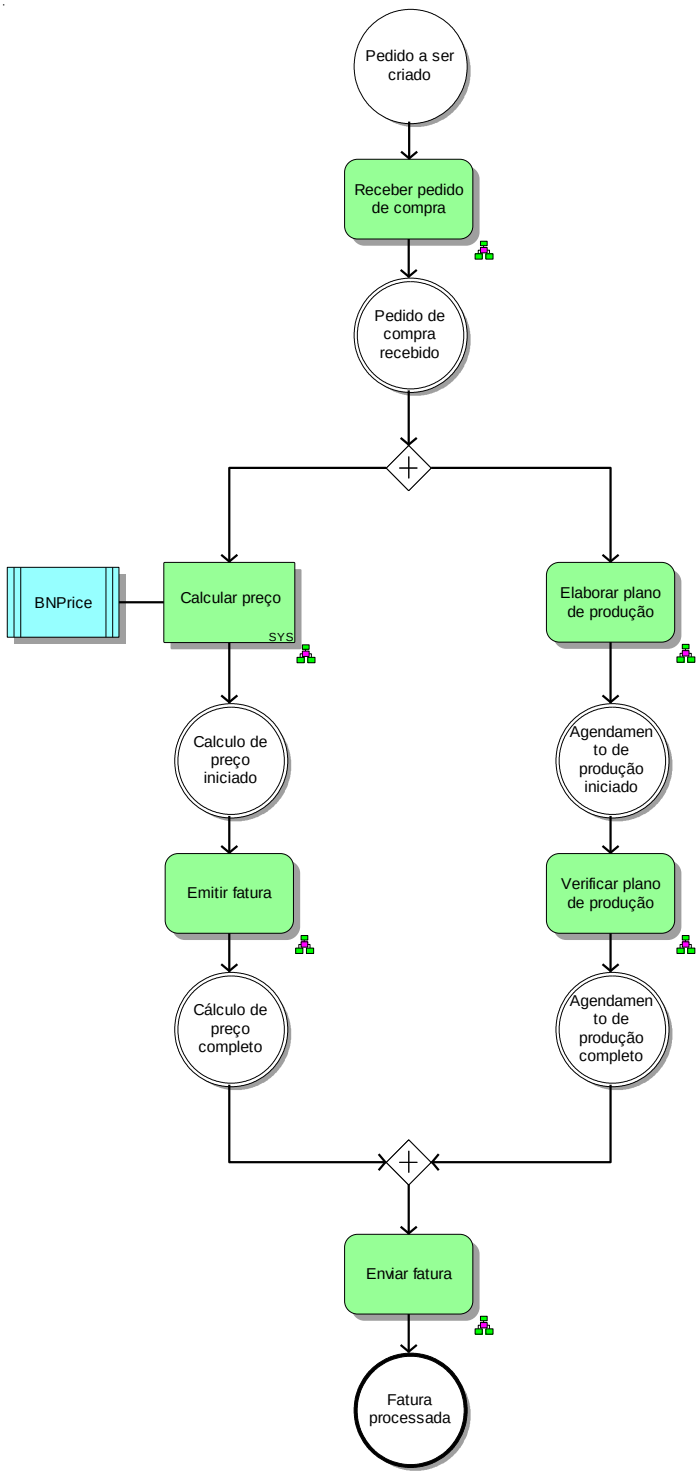


Figura 19 – EPC com a atividade CalcularPreço automatizada

Contudo, existe uma possível inconsistência nesse modelo. Segundo a Figura 13, a atividade Calcular Preço recebe um cluster Pedido de Compra como entrada e tem como saída um cluster Preço. Porém, não dissemos no ARIS qual é o correspondente desses clusters no serviço. Para compatibilizar a mensagem do serviço com o cluster da atividade, criaremos um diagrama de classe como o da Figura 20, ligando o tipo de dado complexo do WSDL com o cluster correspondente.

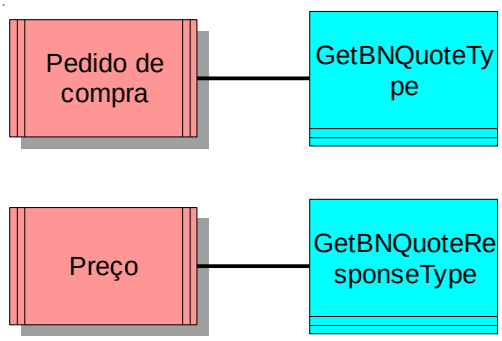


Figura 20 – Diagrama de classe com a correspondência entre *cluster* e tipo de dado complexo

Neste momento, estamos prontos para realizar a transformação do processo de negócio em um processo de TI, ou seja, um serviço orquestrado. Ao ser realizada a transformação, um novo processo BPEL é criado, como mostra a Figura 21.

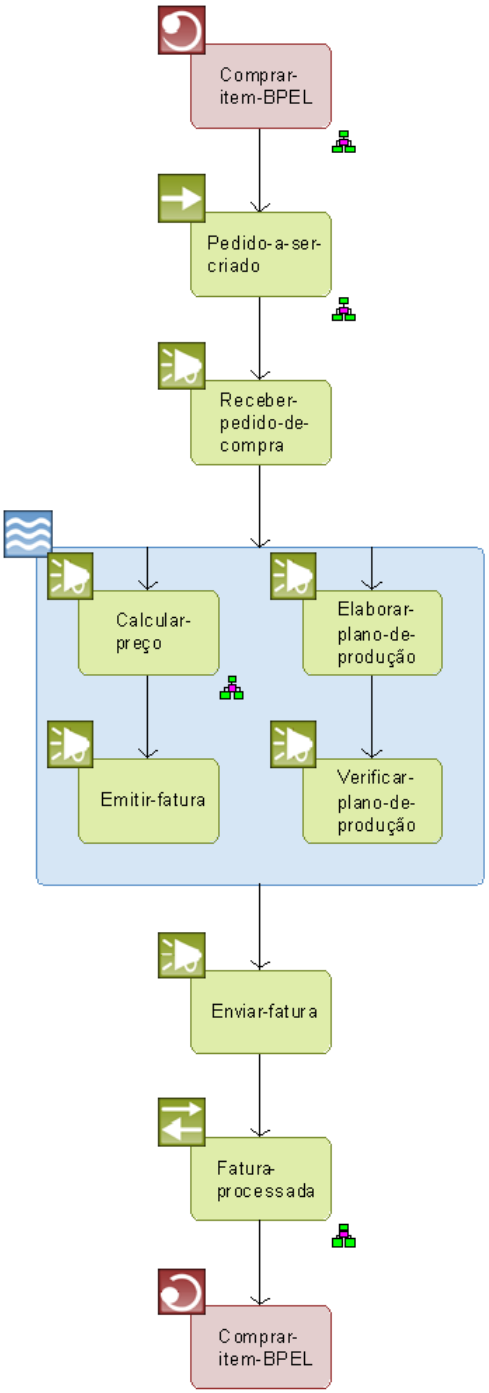


Figura 21 – Diagrama de processo BPEL transformado de um EPC

O processo BPEL gerado possui atividades de invocação, uma atividade que representa o recebimento de uma mensagem de requisição no processo (Pedido-a-ser-criado) e uma atividade que representa o recebimento de uma mensagem de resposta do processo (Fatura-processada). O elemento que denota o início do processo, Comprar-item-BPEL, possui um diagrama de alocação com as definições do processo, como mostra a figura abaixo.

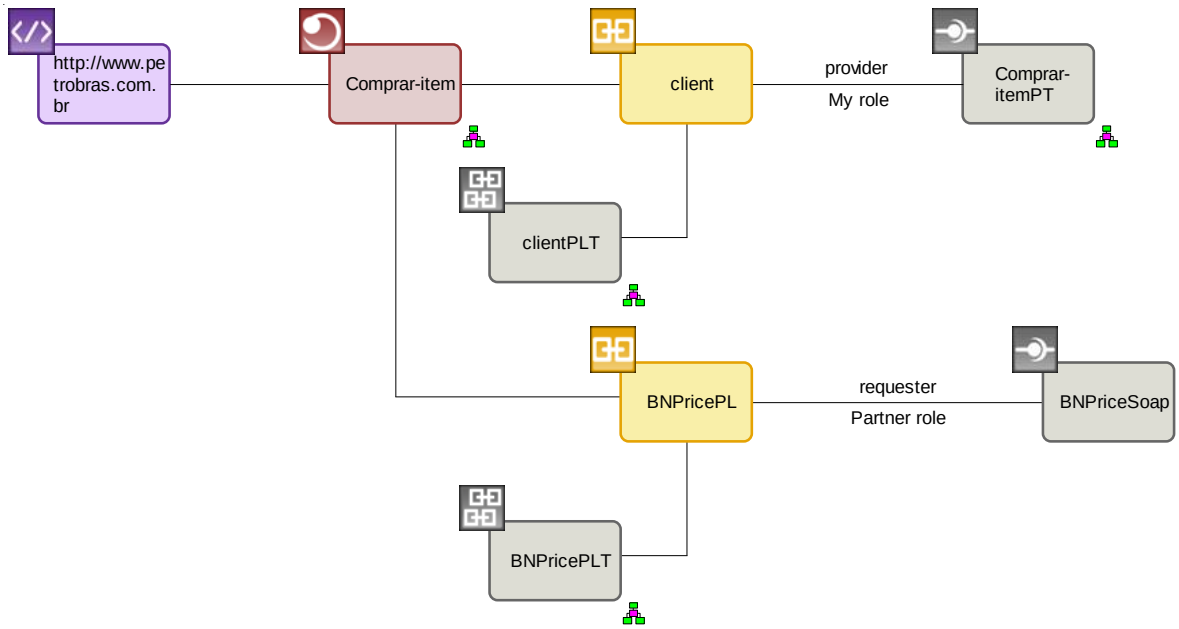


Figura 22 – Diagrama de alocação Comprar-item-BPEL

O diagrama de alocação acima possui a definição do namespace do processo (http://www.petrobras.com.br) e dos dois parceiros que se relacionam no processo, BNPricePL e client. O primeiro diz respeito ao parceiro denominado de BNPricePL, o qual possui um PartnerLinkType BNPricePLT com o PortType BNPriceSoap do serviço BNPrice. Simplificando, essa ligação entre PartnerLink, PartnerLinkType e PortType define um serviço que será utilizado no processo BPEL e as operações (presentes no PortType) que o serviço possui. O segundo, chamado de client, diz respeito ao próprio processo BPEL Comprar-item, o qual possui uma parceria externa. Ou seja, uma vez que, o processo BPEL será chamado por outro serviço ou barramento, a porta de entrada para recebimento de requisições é considerada também um parceiro do processo BPEL.

Ao expandirmos o elemento Comprar-itemPT podemos verificar os detalhes do PortType do processo BPEL, conforme mostra a figura abaixo. O PortType possui uma operação, chamada de executePedido-a-ser-criado(). Assim, quando essa operação for requisitada, o processo BPEL é iniciado.

Figura 23 – PortType Comprar-itemPT

No processo gerado, podemos verificar que cada atividade do EPC foi transformada em uma invocação de serviço. As atividades que não estavam automatizadas não foram relacionadas com nenhum serviço e este relacionamento pode ser feito futuramente. Já a atividade Calcular Preço está detalhada em um diagrama de alocação

exibido na Figura 24, onde está indicado que será chamado o parceiro BNPrice, através de sua operação GetBNQuote do PortType BNPriceSoap. Como o PortType BNPriceSoap possui somente uma operação, essa operação foi automaticamente selecionada. Caso o PortType possua mais de uma operação, essa associação não será feita e é necessário representar manualmente no modelo BPEL a operação a ser invocada pelo processo.

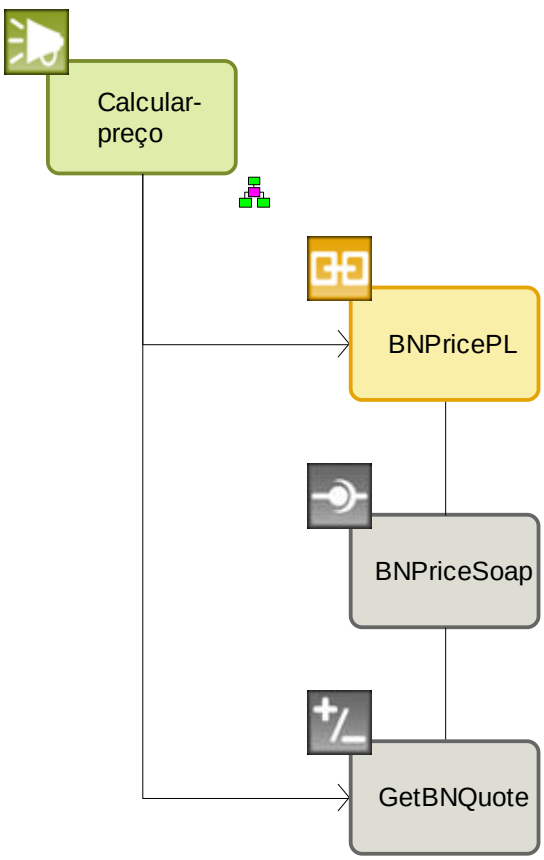


Figura 24 – Diagrama de alocação Calcular Preço

Por fim, as atividades Pedido-a-ser-criado e Fatura-processada representam, respectivamente, uma recepção de mensagem e o envio de uma mensagem de reposta. Ambas são muito parecidas e estão detalhadas nas Figura 25 e Figura 26. A recepção descreve que o parceiro client (o próprio processo BPEL), através da sua operação executePedido-a-ser-criado(), receberá uma mensagem para execução do processo. Por sua vez, o envio da mensagem de resposta descreve que o parceiro client enviará uma resposta através da sua operação executePedido-a-ser-criado().

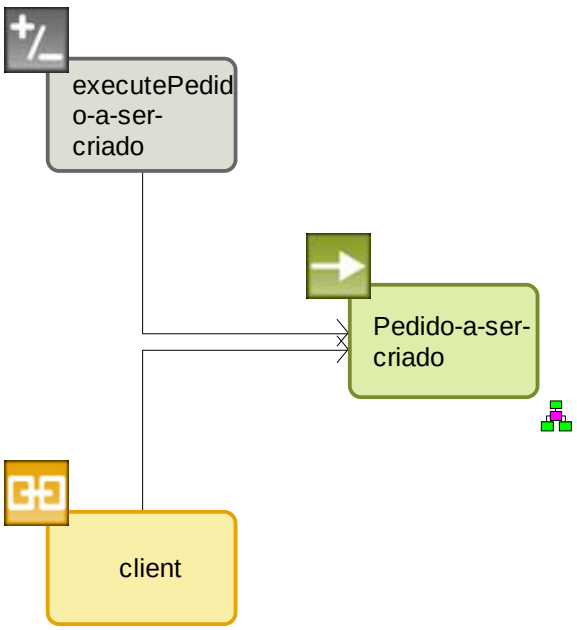


Figura 25 – Diagrama de alocação Pedido-a-ser-criado

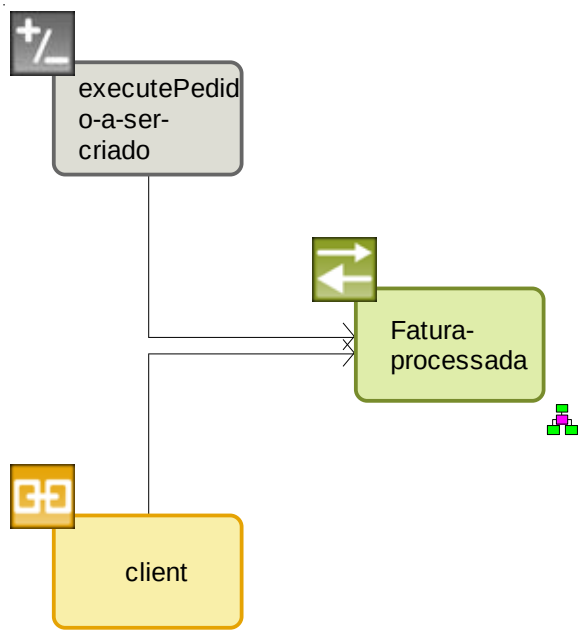


Figura 26 – Diagrama de alocação Fatura-processada

As demais atividades do processo, como não estavam ainda automatizadas, não possuem detalhamento. Porém, o especialista de TI responsável pela orquestração dos serviços pode completar o processo conforme os serviços forem sendo desenvolvidos.

Por último, o processo pode ser exportado para o formato BPEL para ser consumido pelo barramento de serviços. Ao exportamos o processo acima, o ARIS gera os seguintes arquivos: Comprar-item.bpel (descrição do processo), Comprar-item.wsdl (descrição da interface do processo para acesso por outros serviços ou aplicações) e

BNPricePLT.wsdl (definição do mapeamento do parceiro BNPricePLT para o serviço Web BNPrice). Seguem abaixo os códigos gerados.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by the ARIS SOA Architect, IDS Scheer AG. All rights reserved. www.ids-
scheer.com-->
<process                 xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:tns="http://www.petrobras.com.br"     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"    name="Comprar-item"
queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
targetNamespace="http://www.petrobras.com.br">
  <partnerLinks>
    <partnerLink myRole="provider" name="client" partnerLinkType="tns:clientPLT"/>
    <partnerLink            xmlns:imp1="http://www.abundanttech.com/webservices/BNPrice"
name="BNPricePL" partnerLinkType="imp1:BNPricePLT" partnerRole="requester"/>
  </partnerLinks>
  <sequence>
    <receive createInstance="yes" name="Pedido-a-ser-criado" operation="executePedido-a-
ser-criado" partnerLink="client" portType="tns:Comprar-itemPT"/>
    <invoke name="Receber-pedido-de-compra"/>
    <flow name="AND">
      <sequence>
        <invoke name="Plano-de-produção"/>
        <invoke name="Verificar-plano-de-produção"/>
      </sequence>
      <sequence>
        <invoke             xmlns:imp1="http://www.abundanttech.com/webservices/BNPrice"
name="Calcular-preço"          operation="GetBNQuote"          partnerLink="BNPricePL"
portType="imp1:BNPriceSoap"/>
        <invoke name="Emitir-fatura"/>
      </sequence>
    </flow>
    <invoke name="Enviar-fatura"/>
    <reply        name="Fatura-processada"        operation="executePedido-a-ser-criado"
partnerLink="client" portType="tns:Comprar-itemPT"/>
  </sequence>
</process>
```

Figura 27 – Conteúdo do arquivo Comprar-item.bpel

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by the ARIS SOA Architect, IDS Scheer AG. All rights reserved. www.ids-
scheer.com-->
<definitions                                    xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:imp1="http://www.abundanttech.com/webservices/BNPrice"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:tns="http://www.petrobras.com.br"     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
name="Comprar-item" targetNamespace="http://www.petrobras.com.br">
  <portType name="Comprar-itemPT">
    <operation name="executePedido-a-ser-criado"/>
  </portType>
  <plnk:partnerLinkType name="clientPLT">
    <plnk:role name="provider">
      <plnk:portType name="tns:Comprar-itemPT"/>
    </plnk:role>
  </plnk:partnerLinkType>
</definitions>
```

Figura 28 – Conteúdo do arquivo Comprar-item.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by the ARIS SOA Architect, IDS Scheer AG. All rights reserved. www.ids-
scheer.com-->
<definitions                                    xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:impl="http://www.abundanttech.com/webservices/BNPrice"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"                              name="BNPricePLT"
targetNamespace="http://www.abundanttech.com/webservices/BNPrice">
  <import   location="file:/T:/Documents%20and%20Settings/BC7D/Desktop/bnprice_soap.wsdl"
namespace="http://www.abundanttech.com/webservices/BNPrice"/>
  <plnk:partnerLinkType name="BNPricePLT">
    <plnk:role name="requester">
      <plnk:portType                                          name="impl:BNPriceSoap"
xmlns:impl="http://www.abundanttech.com/webservices/BNPrice"/>
    </plnk:role>
  </plnk:partnerLinkType>
</definitions>
```

Figura 29 – Conteúdo do arquivo BNPricePLT.wsdl

Analisando o processo BPEL criado, verificamos que a correspondência entre os clusters e os tipos de mensagens, feito na Figura 20, não foi levado em consideração na transformação. De fato, toda a informação presente em diagramas FAD não foi utilizada na transformação e, assim, essas informações terão que ser inseridas manualmente pelo usuário no diagrama de processo BPEL.

3.2 Exemplo 2: utilizando o máximo de informação do diagrama EPC

Este segundo exemplo conterá os mesmos passos do primeiro exemplo, porém uma única mudança: a modelagem do EPC orientado a serviços. Neste exemplo, adicionaremos ao processo da Figura 11 todos os elementos presentes nos FADs das atividades que compõem o processo. Assim, permitiremos que o módulo SOA possa interpretar todas as informações modeladas do processo. Contudo, como sabemos, essa forma de modelagem não é permitida pelas diretrizes atuais do E&P [E&P, 2008]. Assim, após selecionarmos o serviço BNPrice e adicionarmos no EPC as informações no FAD, o diagrama ficará como mostra a Figura 30.

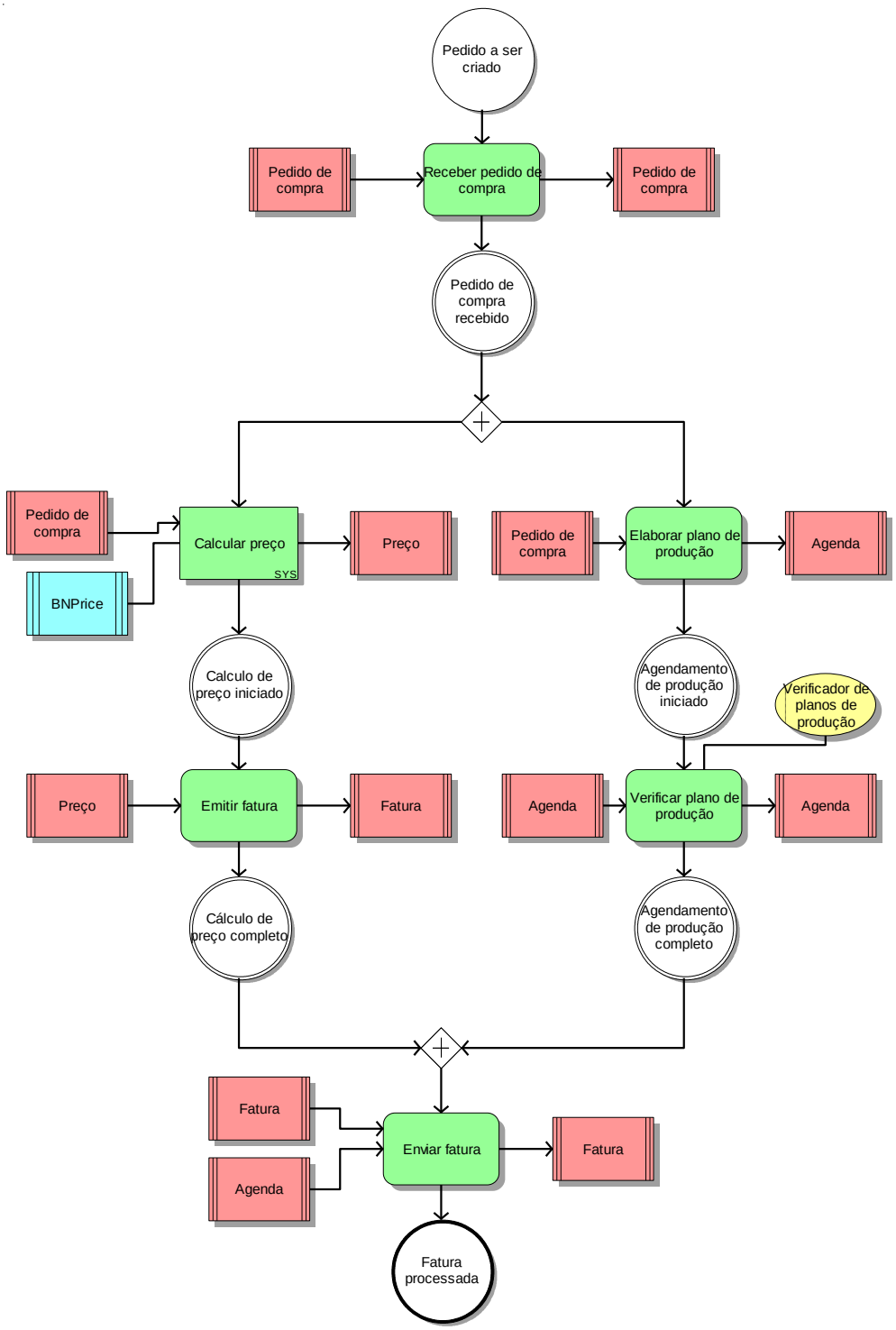


Figura 30 – EPC com elementos do FAD copiados para o EPC

Após acrescentar os elementos no EPC, o módulo SOA será capaz de utilizar os elementos de entrada e saída de infomações *(clusters)* para definir as mensagens de entrada e saída das atividades do processo BPEL. Após a transformação do diagrama EPC em um diagrama de processo BPEL, temos o diagrama abaixo:

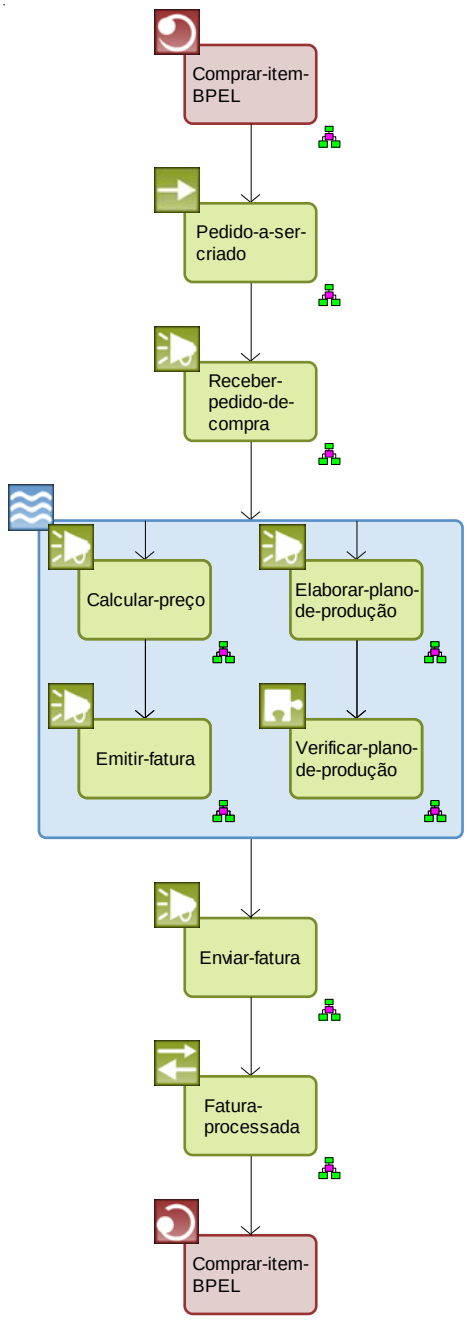


Figura 31 – Segundo diagrama de processo BPEL transformado de um EPC

Analisando o fluxo de atividades BPEL, podemos reparar que o diagrama gerado possui duas diferenças visíveis em comparação com o diagrama gerado no exemplo anterior. Em primeiro lugar, temos uma diferença estrutural visível: a atividade Verificar-plano-de-produção, anteriormente representada como uma atividade de invocação, agora está representada como uma extensão (vide seção A.1.5). Tal mudança de representação se dá por causa do objeto Organizational Unit Type "Verificador de planos de produção" que, para o módulo SOA, representa um módulo do Enterprise Service Bus no qual o processo será disponibilizado que possui uma funcionalidade específica que será chamada pela atividade "Verificar plano de produção". Além da mudança estrutural do processo com a atividade de extensão,

podemos verificar que todas as atividades BPEL estão associadas a diagramas de alocação BPEL. De fato, os diagramas de alocação são utilizados para detalhar as atividades do processo e, uma vez que colocamos mais informações no diagrama EPC, essas informações estão detalhadas em diagramas de alocação.

Primeiro, analisaremos o diagrama de alocação Comprar-item-BPEL que possui as definições do processo BPEL. O diagrama de alocação pode ser visualizado na Figura 32. A principal diferença do diagrama em comparação com o diagrama correspondente do exemplo anterior é a presenta das definições das variáveis do processo (denotadas com o símbolo $a). Duas dessas variáveis, "Pedido-de-compra" e "Preço", foram mapeadas para os tipos de dados correspondentes ("GetBNQuoteType" e "GetBNQuoteResponseType", respectivamente), conforme definimos na Figura 20.

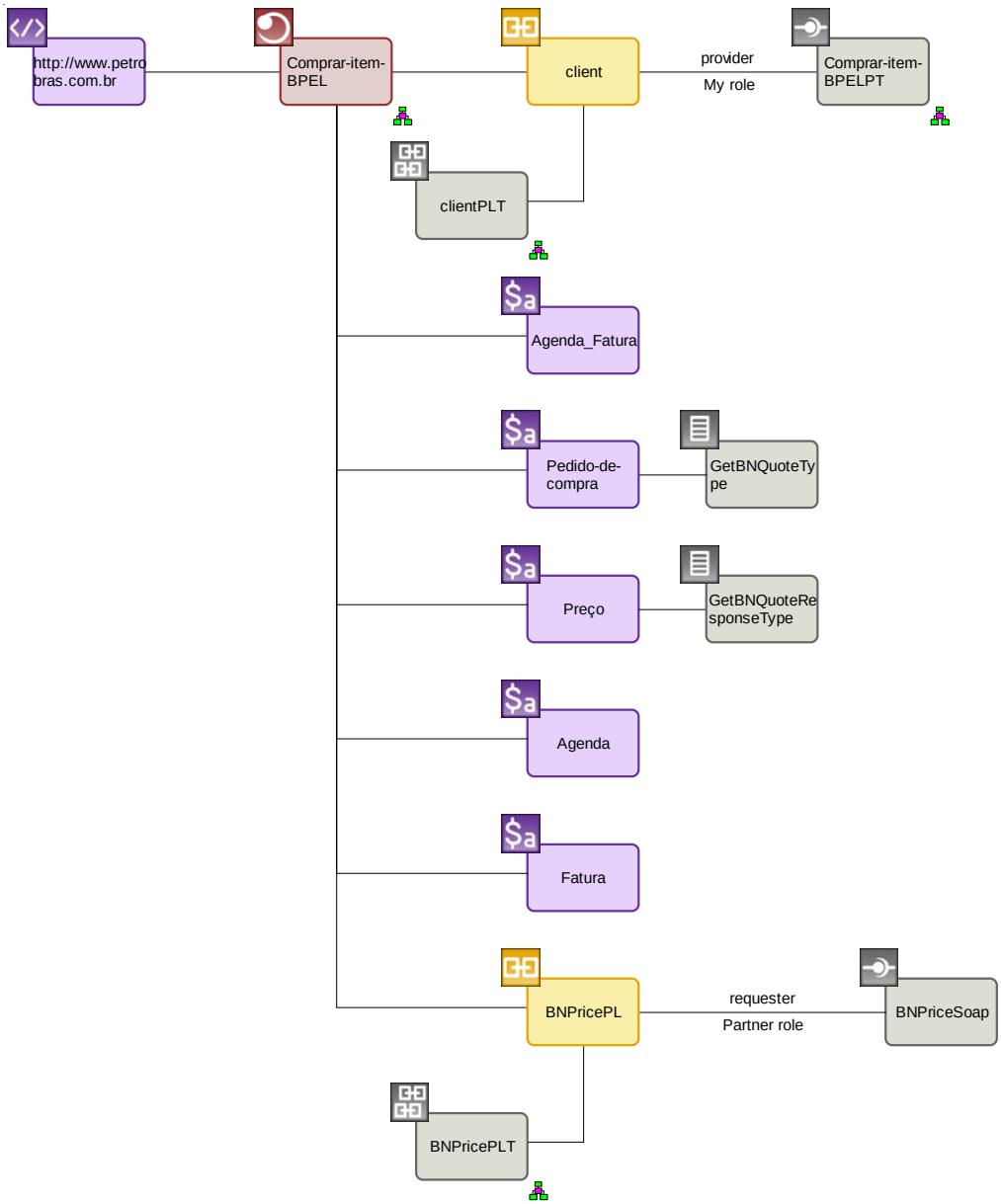


Figura 32 – Diagrama de alocação Comprar-item-BPEL

As variáveis definidas no diagrama de alocação acima são utilizadas nos diagramas de alocação das atividades que possuem entradas e saídas de mensagens. Para conhecermos todas as informações presentes no diagrama BPEL, colocamos abaixo os demais diagramas de alocação que foram gerados nesta transformação.

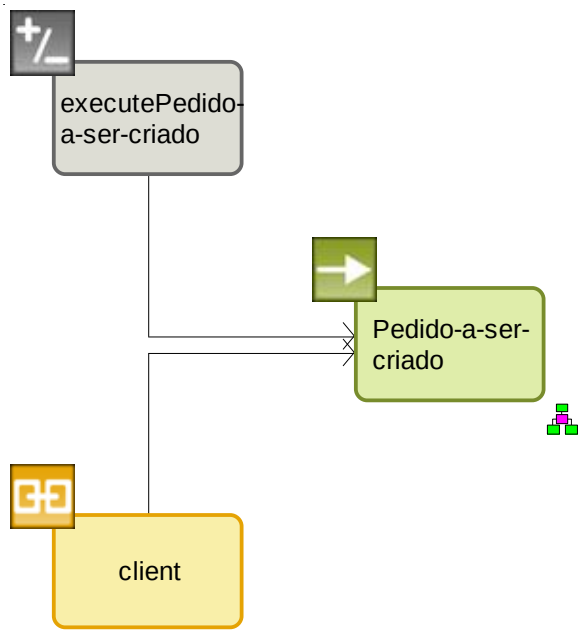


Figura 33 – Diagrama de alocação Pedido-a-ser-criado

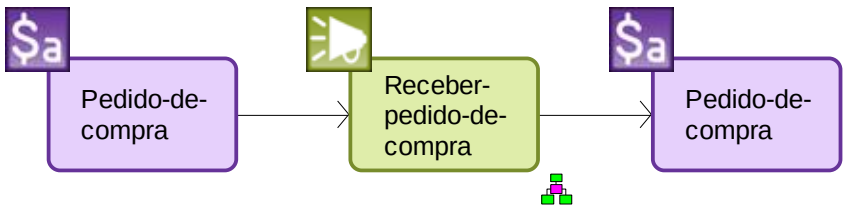


Figura 34 – Diagrama de alocação Receber-pedido-de-compra

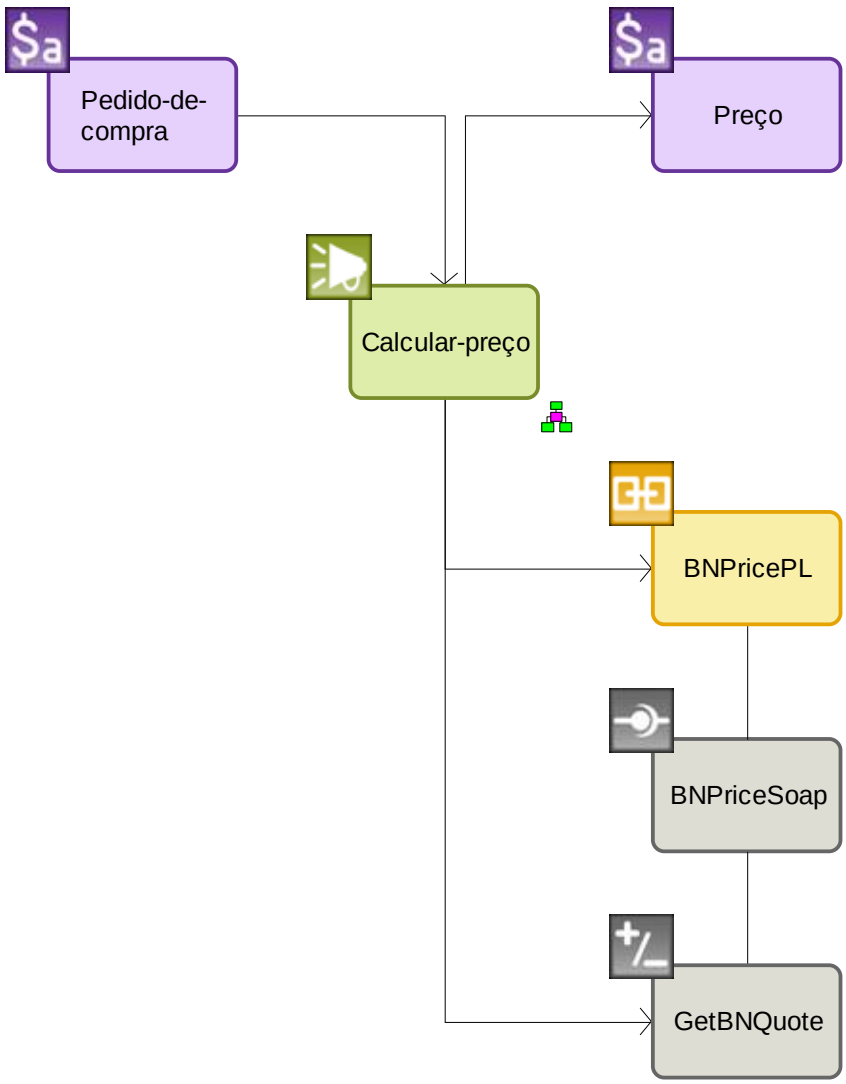


Figura 35 – Diagrama de alocação Calcular-preço

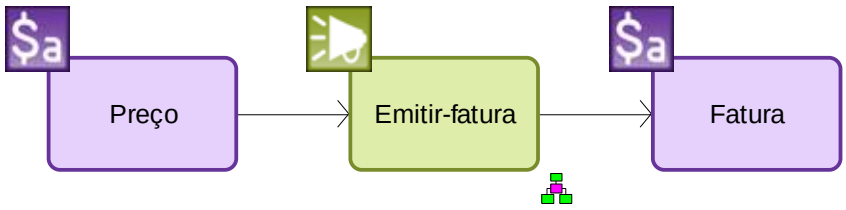


Figura 36 – Diagrama de alocação Emitir-fatura

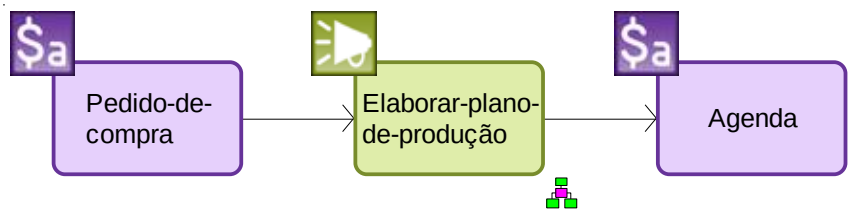


Figura 37 – Diagrama de alocação Elaborar-plano-de-produção

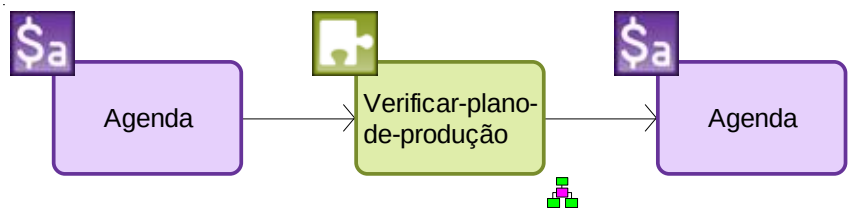


Figura 38 – Diagrama de alocação Verificar-plano-de-produção

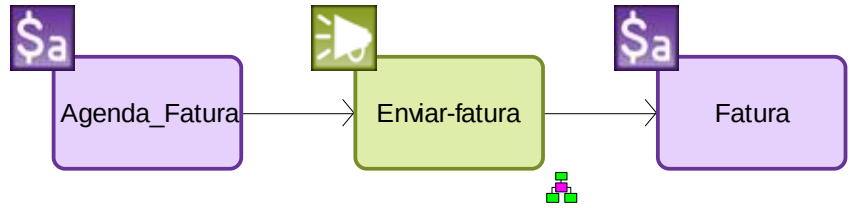


Figura 39 – Diagrama de alocação Enviar-fatura

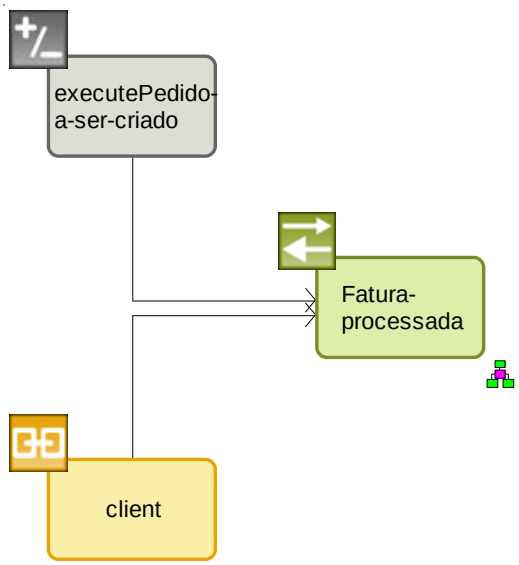


Figura 40 – Diagrama de alocação Fatura-processada

Adicionalmente, foram criadas também a visão funcional e a visão técnica do processo, conforme pode ser visualizado na Figura 41 e na Figura 42, respectivamente.

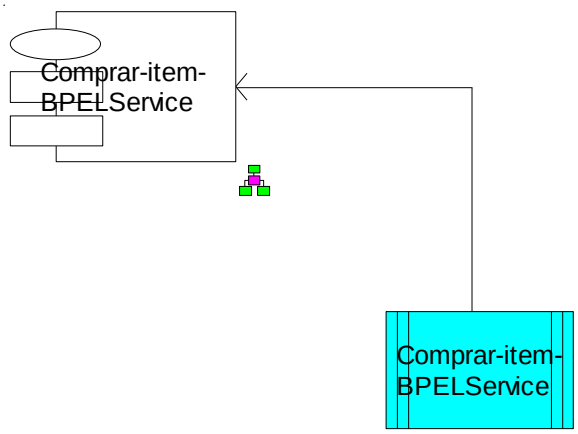


Figura 41 – Visão funcional do processo Comprar-Item-BPEL

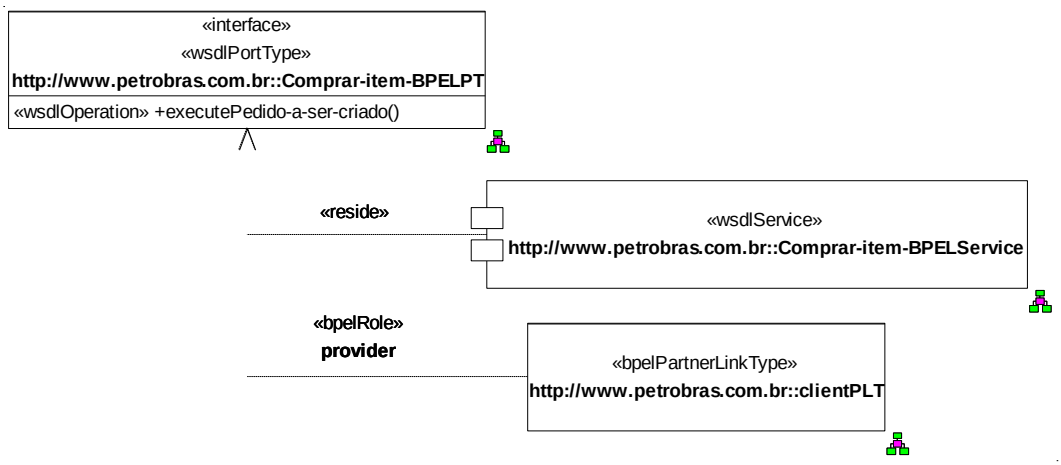


Figura 42 – Visão técnica do processo Comprar-item-BPEL

Por último, o processo pode ser novamente exportado para o formato BPEL para ser consumido pelo barramento de serviços. A exportação irá gerar os memos arquivos gerados na exportação do exemplo anterior. Porém, uma vez que o processo BPEL foi substancialmente diferente neste exemplo, o conteúdo do arquivo Comprar-item.bpel (descrição do processo) será igualmente distinto. Segue abaixo o código do arquivo Comprar-item.bpel. Note a presença da declaração das variáveis (`variable`) e da atividade de extensão (`bpelx`) no código.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by the ARIS SOA Architect, IDS Scheer AG. All rights reserved. www.ids-
scheer.com-->
<process                  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:tns="http://www.petrobras.com.br"      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"     name="Comprar-item-
BPEL"                         queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
targetNamespace="http://www.petrobras.com.br">
  <partnerLinks>
    <partnerLink             xmlns:imp1="http://www.abundanttech.com/webservices/BNPrice"
name="BNPricePL" partnerLinkType="imp1:BNPricePLT" partnerRole="requester"/>
    <partnerLink myRole="provider" name="client" partnerLinkType="tns:clientPLT"/>
  </partnerLinks>
  <variables>
    <variable name="Preço" type="GetBNQuoteResponseType"/>
    <variable name="Agenda"/>
    <variable name="Agenda_Fatura"/>
    <variable name="Fatura"/>
    <variable name="Pedido-de-compra" type="GetBNQuoteType"/>
  </variables>
  <sequence>
    <receive createInstance="yes" name="Pedido-a-ser-criado" operation="executePedido-a-
ser-criado" partnerLink="client" portType="tns:Comprar-item-BPELPT"/>
    <invoke       inputVariable="Pedido-de-compra"       name="Receber-pedido-de-compra"
outputVariable="Pedido-de-compra"/>
    <flow name="AND">
      <sequence>
        <invoke              xmlns:imp1="http://www.abundanttech.com/webservices/BNPrice"
inputVariable="Pedido-de-compra"       name="Calcular-preço"       operation="GetBNQuote"
outputVariable="Preço" partnerLink="BNPricePL" portType="imp1:BNPriceSoap"/>
        <invoke inputVariable="Preço" name="Emitir-fatura" outputVariable="Fatura"/>
      </sequence>
      <sequence>
        <invoke    inputVariable="Pedido-de-compra"    name="Elaborar-plano-de-produção"
outputVariable="Agenda"/>
        <bpelx          inputVariable="Agenda"         name="Verificar-plano-de-produção"
outputVariable="Agenda"/>
      </sequence>
    </flow>
    <invoke inputVariable="Agenda_Fatura" name="Enviar-fatura" outputVariable="Fatura"/>
    <reply        name="Fatura-processada"        operation="executePedido-a-ser-criado"
partnerLink="client" portType="tns:Comprar-item-BPELPT"/>
  </sequence>
</process>
```

Figura 43 – Conteúdo do arquivo Comprar-item.bpel

4 Avaliação do uso da ferramenta no projeto

O módulo SOA da ferramenta ARIS fornece uma notação gráfica para os elementos BPEL e WSDL, facilitando a compreensão em alto nível da composição de serviços. Além disso, a ferramenta pode se tornar um repositório documental dos serviços disponíveis na organização por armazenar as interfaces WSDL e os tipos de dados XSD, e do relacionamento desses serviços com os processos de negócio modelados.

O módulo pode ser utilizado para modelagem dos processos de TI, ou seja, a orquestração de serviços em linguagem BPEL. A principal característica do módulo, frente a outras ferramentas do mercado, é a ligação entre os processos BPEL e a modelagem dos processos de negócio (BPMN) realizada em ambiente ARIS, o qual é utilizado na E&P. Porém, a transformação de elementos EPC em diagramas de processo BPEL não pode ser feita de forma automática utilizando todas as informações disponíveis em um processo modelado utilizando as diretrizes de modelagem da E&P [E&P, 2008], uma vez que o módulo SOA do ARIS não se utiliza dos elementos

presentes em FADs. Além disso, essa transformação pode não ser completa, uma vez que informações importantes para o processo automatizado BPEL não estão disponíveis no diagrama de processo de negócio EPC. Um exemplo de informação ausente no EPC é o detalhamento de qual operação de um serviço dá apoio a certa atividade. Uma vez que essa informação não é representada no EPC, a transformação para processo BPEL somente representa invocações de operações de serviços que possuam somente uma única operação. Caso o serviço possua mais de uma operação, a invocação da operação deve ser feita manualmente, após a transformação, o que gera trabalho adicional do modelador.

Mesmo podendo ser utilizado como um repositório, o módulo em si não fornece quaisquer funcionalidades avançadas para catalogação dos serviços. É possível realizar uma busca por serviços, porém esta é uma busca simples por elemento e nome do elemento. Não existe a opção para cadastrar serviços com UDDI, por exemplo. Além disso, o módulo SOA não disponibiliza opções para monitoramento dos processos, porém, é possível que essa funcionalidade venha a ser atendida por um módulo do ARIS específico para monitoramento de processos, como o ARIS Process Event Monitor. Ainda, o módulo é compatível com a especificação da linguagem 1.1 do BPEL e 1.1 do WSDL e não com as especificações mais novas dessas linguagens: 2.0 do BPEL, também chamada de WS-BPEL [WS-BPEL, 2007] e 2.0 do WSDL [WSDL, 2007].

Glossário

Protocolo de negócio - "Business protocol" é um conceito do da especificação BPEL. O sinônimo de "protocolo de negócio" é "processo abstrato". Ou seja, é um processo que descreve o comportamento na troca de mensagens dos parceiros de negócio e que é visível para todos os parceiros. Cada parceiro descreve sua parte da interação, os processos internos da interação continuam escondidos.

XML Schema – XML Schema é uma linguagem que é usada para descrever a estrutura de um documento XML. Um documento XML Schema muitas vezes é utilizado para validar um documento XML.

XML Schema Definitions – Definições de XML Schema (ou XSD) é uma instância de um XML Schema descrito em XML Schema. Um XSD define um tipo de documento XML em termos de restrições em que os elementos e atributos podem aparecer, seus relacionamentos, que tipos de dados podem ter, entre outros. Por exemplo, um XSD pode definir o elemento Casa e Locatário e dizer que cada elemento Locatário deve possuir no mínimo 1 elemento Casa associado a ele. Além disso, o elemento Casa deve possuir um atributo valorDoAluguel.

URI (Uniform Resource Identifier) – Um conjunto de caracteres usado para identificar ou nomear um recurso. As URI's podem ser classificadas em URN's e URL's. Um URN (Uniform Resource Name) funciona como o nome de uma pessoa e uma URL (Uniform Resource Locator) funciona como o endereço dessa pessoa. Um URN conhecido é o ISBN utilizado para identificar livros. Uma URL é o caminho para se chegar ao livro (pode ser o caminho de um site ou o caminho de um arquivo).

Referências

BPEL4WS, **Business Process Execution Language for Web Services (BPEL4WS) 1.1**, 2003. disponível em http://www-106.ibm.com/developerworks/webservices/library/ws-bpel.

E&P, **Diretrizes para modelagem de processos na E&P versão 8.0**, Petrobras, 2008.

IDS Scheer, **ARIS SOA Designer 7.0 Manual**, versão 1.0, outubro de 2006.

WS-BPEL, **Web Services Business Process Language (WS-BPEL) 2.0**, 2007, disponível em http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html#_Toc164738489.

WSDL, **Web Services Description Language (WSDL) 1.1**, 2001, disponível em www.w3.org/TR/wsdl.

WSDL, **Web Services Description Language (WSDL) 2.0**, 2007, disponível em http://www.w3.org/TR/wsdl20/.

XSD, **XML Schema Part 1: Structures Second Edition**, 2004, disponível em http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/#components.

Apêndice A - Notação BPEL utilizada no módulo ARIS

Esse anexo detalha os elementos utilizados na modelagem para o desenvolvimento de serviços. Num primeiro momento, é apresentada a notação dos elementos BPEL suportados pelo módulo ARIS e a sua semântica. Em seguida, são apresentados os elementos de modelagem de negócio que são utilizados para ajudar na criação de diagramas BPEL.

A.1.1Símbolos para representação de elementos BPEL

A tabela 1 apresenta os símbolos utilizados pelo ARIS para representar os elementos BPEL.

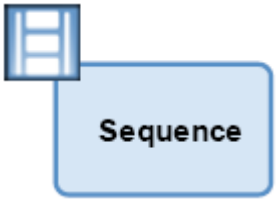
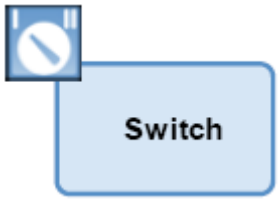
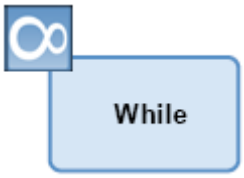
Tabela 1. Símbolos BPEL utilizados pelo ARIS

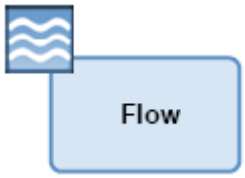
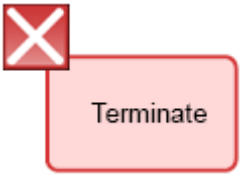
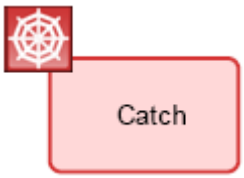
Elemento	Descrição	Referência
 **Figura 44 – Atividade Invoke básica**	Invoke é usado para invocar um serviço utilizando uma operação one-way ou request/response. Um Invoke que chama uma operação do tipo one-way precisa apenas de uma variável de entrada, pois ela não espera um valor de retorno (chamada assíncrona). Um Invoke que chama uma operação do tipo request/response precisa de uma variável de entrada e uma de saída (chamada síncrona).	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 11.3.

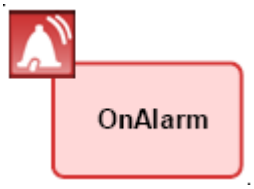
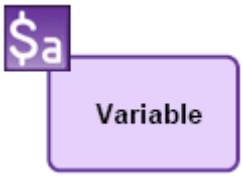
Elemento	Descrição	Referência
 **Figura 45 – Atividade Receive básica**	Receive é usado se o processo espera uma mensagem de entrada específica. PartnerLink (vide Figura 68), PortType (vide Figura 74) e Operation (vide Figura 75) de onde a mensagem é esperada são incluídos junto com o Receive para especificar qual a mensagem que é esperada. Se o atributo createInstance do elemento Receive estiver marcado pra "sim", o Receive deve ser modelado como atividade inicial no processo de negócio. Atividades iniciais simultâneas são permitidas, ou seja, o processo pode possuir mais de um recebimento de mensagem como atividade inicial.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 11.4.
 **Figura 46 – Atividade Reply básica**	Reply é usado para indicar uma resposta para uma mensagem que foi recebida anteriormente usando Receive. Uma combinação de Receive/Reply representa uma operação request/response. Nos processos executáveis, uma variável contendo a mensagem de texto recebida/enviada deve ser incluída. Uma consulta sobre um ParnertLink específico (vide Figura 68), PortType (vide Figura 74), Operation (vide Figura 75) e CorrelationSet (vide Figura 66) deve sempre ser respondida antes da nova consulta ser feita.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 11.4.

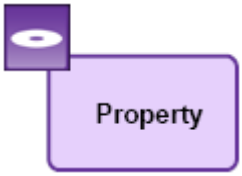
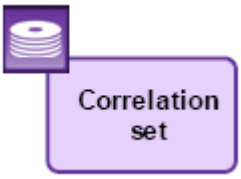
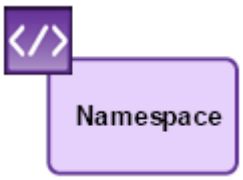
Elemento	Descrição	Referência
 **Figura 47 – Atividade Assign básica**	Assign é usado para atualizar os valores das variáveis,tanto quando novos valores são calculados como quando valores são copiados de uma variável para outra de tipo compatível.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 9.3.
 **Figura 48 – Atividade Copy básica**	Copy é usado junto com a atividade básica Assign e representa a cópia de elemento XML. Ao copiar valores de uma variável pra outra, você deve ter certeza de que os tipos são compatíveis.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 9.3.
 **Figura 49 – Atividade Wait básica**	Wait é usado para invocar uma operação em um momento específico ou para parar um processo por um determinado período.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 11.7.
 **Figura 50 – Atividade Empty básica**	Empty é uma atividade que representa uma atividade sem ação. Em termos práticos, é uma atividade que não faz nada. Ela é requerida se erros forem identificados com um elemento Throw, porém esses erros não quiserem ser tratados, por exemplo.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 11.8.

Elemento	Descrição	Referência
 **Figura 51 – Atividade Compensate**	Compensate pode ser usado se todas as mudanças que podem ser atribuídas a atividades de um escopo anterior, normalmente completo, tiverem que ser revertidas. O elemento Compensate emite um pedido de compensação para o processo ou para um escopo. Uma compensação pode ser disparada após o lançamento de algum evento (como um evento de erro, por exemplo). A compensação é utilizada sobre atividades que terminaram corretamente e, por algum evento lançado durante o processo, necessitam ser compensadas (ou seja, serem revertidas ou atividades adicionais serem executadas).	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 11.3.2.
 **Figura 52 – Atividade Extension ARIS BPEL**	Um Extension representa uma extensão do padrão BPEL específica de uma determinada plataforma. A extensão do BPEL é feita por algumas plataformas como IBM, Microsoft, Oracle, etc, que provêm soluções para problemas de desempenho em transformação de dados ou cálculos muito complexos que são necessários para preparar os dados a serem usados por um serviço web invocado. Este elemento permite que estas extensões implementadas por outras ferramentas possam ser utilizadas.	Elemento BPEL.

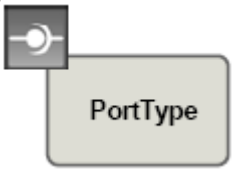
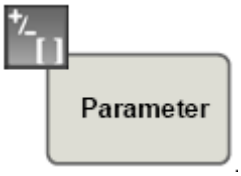
Elemento	Descrição	Referência
 **Figura 53 – Atividade estruturada Sequence**	Sequence contém uma ou mais atividades que são executadas na ordem na qual estão listadas na seqüência.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 12.1.
 **Figura 54 – Atividade estruturada Switch**	Switch contém uma lista com uma ou mais condições if-then. Dependendo se a condição for atendida, o ramo correspondente é processado.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 12.2.
 **Figura 55 – Atividade estruturada While**	While executa uma atividade específica enquanto uma determinada condição ocorrer.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 12.3.
 **Figura 56 – Atividade estruturada Pick**	Pick espera pela ocorrência de um evento específico dentro de um conjunto determinado e então executa uma atividade correspondente. Se mais de um dos eventos esperados acontecerem, a atividade a ser executada vai depender do evento que ocorreu primeiro.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 12.4.

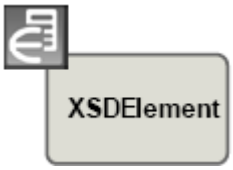
Elemento	Descrição	Referência
 **Figura 57 – Atividade estruturada Flow**	Flow representa elementos que são executados em paralelo. Os elementos que estiverem dentro da área de um elemento Flow são executados ao mesmo tempo. Um Flow estará completo quando todos os elementos forem executados.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 12.5.
 **Figura 58 – Atividade básica Throw**	Throw é utilizado se um erro interno é sinalizado explicitamente. A atividade Throw usa o nome único do erro e pode também incluir uma variável contendo informações sobre o erro.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 11.6.
 **Figura 59 – Atividade básica Terminate**	Terminate é usado se a execução do processo tiver de ser cancelada imediatamente. Todas as atividades sendo executadas no momento são canceladas o mais rápido possível sem nenhum tratamento padrão e sem ação de reset. A atividade Terminate só é permitida em processos executáveis.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 14.6.
 **Figura 60 – Elemento ARIS BPEL Catch**	Catch representa uma definição de um tratamento padrão dentro de um processo BPEL. Catch é usado para capturar um erro específico que pode ser o resultado de uma operação. O gatilho do erro pode ser associado ao objeto Catch.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 13.4.

Elemento	Descrição	Referência
 **Figura 61 – WSDL erro Fault**	Fault é um elemento opcional e especifica o formato da mensagem abstrata para um erro, o qual pode ser o resultado de uma operação. Todo erro deve ter um nome de qualificação único.	Elemento WSDL. Mais informações sobre a semântica desse elemento podem ser encontradas em [WSDL, 2001], seção 6.1.
 **Figura 62 – Elemento ARIS BPEL OnAlarm**	OnAlarm representa uma parte da definição de um tratamento de evento dentro de um processo BPEL. OnAlarm indica um evento de *timeout*, ou seja, um elemento que não retornou uma resposta no tempo esperado pode gerar um alarme para o processo.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 13.5.2.
 **Figura 63 – Elemento ARIS BPEL OnMessage**	OnMessage representa uma parte da definição de um tratamento de evento dentro de um processo BPEL. OnMessage é usado para indicar que o tratamento do evento espera por uma mensagem de entrada específica. Essa tag é interpretada de maneira similar a de uma atividade Receive.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 13.5.1.
 **Figura 64 – Elemento Variable**	Variable provê mensagens contendo a situação de um processo de negócio. Variable pode também conter dados que são enviados ou recebidos de parceiros. Uma Variable é definida dentro de um escopo ou processo e seu nome deve ser único. Variable do tipo global é parte do escopo de processo global, enquanto Variable do tipo local são parte do escopo não-global.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 9.2.

Elemento	Descrição	Referência
 **Figura 65 – Elemento Property**	Property representa um campo dentro de uma mensagem e que pode ser acessado através de consultas.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 10.
 **Figura 66 – Correlation set**	Correlation set empacota um grupo nomeado de elementos Property que permitem que a comunicação em nível de aplicação seja identificada dentro de uma instância de protocolo de negócio. Correlation set global se aplica a todo o processo de negócio. Correlation set local é usado apenas dentro de um escopo.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 10.2.
 **Figura 67 – Elemento Namespace**	Namespace representa um Namespace XML. Namespace XML é uma coleção de nomes que são identificados por uma referência URI (RFC2396) e são usadas em documentos XML.	Elemento XML. Mais informações sobre a semântica desse elemento podem ser encontradas em http://www.w3.org/TR/ 1999/REC-xml-names-19990114/.
 **Figura 68 – PartnerLink**	Os serviços com os quais um processo de negócio interage são modelados como parceiros, chamados de PartnerLink em BPEL. Cada PartnerLink é descrito por seu PartnerLinkType, que é a sua porta de entrada. Cada PartnerLink tem seu próprio nome, o qual é usado para todas as interações com o serviço representado pelo PartnerLink.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 7.2.

Elemento	Descrição	Referência
 **Figura 69 – Partner**	Um Partner combina um conjunto de PartnerLinks que definem o relacionamento entre dois processos parceiros. Partner não devem ser sobrepostos, ou seja, um PartnerLink só pode ser usado em um Partner. As definições Partner são opcionais e não devem incluir todos os PartnerLinks usados em um processo de negócio.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 7.3.
 **Figura 70 – ProcessStart**	ProcessStart é um elemento ARIS BPEL que representa o nó inicial de um processo BPEL.	Elemento ARIS.
 **Figura 71 – ProcessEnd**	ProcessEnd é um elemento ARIS BPEL que representa o nó final de um processo BPEL.	Elemento ARIS.
 **Figura 72 – ScopeStart**	ScopeStart é um elemento ARIS BPEL que representa o nó inicial de um escopo dentro de um processo BPEL. Como o ProcessStart, cada ScopeStart pode ser associado a Variables, Correlation Sets, tratamento padrão, etc.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 13.

Elemento	Descrição	Referência
 **Figura 73 – ScopeEnd**	ScopeEnd é um elemento ARIS BPEL que representa o nó final de um escopo dentro de um processo BPEL.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 13.
 **Figura 74 – PortType**	PortType representa um PortType WSDL, ou seja, um conjunto de operações abstratas e a mensagem abstrata associada. O nome usado no atributo name deve ser único. A WSDL possui quatro tipos de transmissões básicas: One-way, resquest-response, solicit-response e notification.	Elemento WSDL. Mais informações sobre a semântica desse elemento podem ser encontradas em [WSDL, 2001], seção 2.4.
 **Figura 75 – Operation**	Operation é uma operação de um PortType do WSDL. Um Operation é identificado por seu atributo nome.	Elemento WSDL. Mais informações sobre a semântica desse elemento podem ser encontradas em [WSDL, 2001], seção 2.4.
 **Figura 76 – Parameter**	Uma lista de Parameter pode ser definida dentro de uma operação. Elas são usadas por Operation com conexão RPC, por exemplo, para capturar a assinatura original de uma função RPC.	Elemento WSDL. Mais informações sobre a semântica desse elemento podem ser encontradas em [WSDL, 2001], seção 2.4.
 **Figura 77 – MessageType**	Uma mensagem consiste de uma ou mais partes, representadas como MessagePart (vide Figura 78).	Elemento WSDL. Mais informações sobre a semântica desse elemento podem ser encontradas em [WSDL, 2001], seção 2.3.

Elemento	Descrição	Referência
 **Figura 78 – MessagePart**	MessagePart é um mecanismo para descrever o conteúdo lógico abstrato de uma mensagem, ou seja, quais os seus elementos e seus tipos. As mensagens são formadas por elementos e seus tipos. Por exemplo, o elemento "nome" pode ser do tipo "string".	Elemento WSDL. Mais informações sobre a semântica desse elemento podem ser encontradas em [WSDL, 2001], seção 2.3.1.
 **Figura 79 – Elemento XSDType**	XSDType representa um tipo de dado descrito em XML Schema, como um xsd:string ou um tipo complexo definido, como Pedido, por exemplo.	Elemento XSD. Mais informações sobre a semântica desse elemento podem ser encontradas em http://www.w3.org/TR /2004/REC-xmlschema-1-20041028/#components.
 **Figura 80 – Elemento XSDElement**	XSDElement representa um elemento XSD. Tipos de dados complexos são criados com elementos. Por exemplo, um tipo de dado complexto Pedido pode conter dois elementos (campos): uma cadeia de caracteres que representa o nome do cliente e um inteiro que representa o número do pedido.	Elemento XSD. Mais informações sobre a semântica desse elemento podem ser encontradas em http://www.w3.org/TR /2004/REC-xmlschema-1-20041028/#components.
 **Figura 81 – PartnerLinkType**	Um PartnerLinkType declara como duas partes envolvidas em uma comunicação interagem uma com a outra e o que cada parte oferece (vide seção A.1.4.2). Ou seja, um PartnerLinkType vai expor quais operações de um dado parceiro (serviço em WSDL) o processo utilizará.	Elemento BPEL. Mais informações sobre a semântica desse elemento podem ser encontradas em [BPEL4WS, 2003], seção 7.1, e em http://www.w3.org/TR /2004/REC-xmlschema-1-20041028/#components.

A.1.2 Associações permitidas para um diagrama de alocação BPEL

Alguns elementos de um diagrama de processo BPEL podem ser associados a um diagrama de alocação que detalhará o elemento, evitando, assim, que o diagrama de processo BPEL se torne muito complexo de ser interpretado caso seja incluídos muitos elementos de detalhamento no diagrama.

Somente os seguintes elementos abaixo podem ser associados a um diagrama de alocação de BPEL:

- Activities (atividades básicas e estruturadas)
- ProcessStart
- ScopeStart
- onMessage
- onAlarm
- Extension activity
- CorrelationSet
- Catch
- PortType
- MessageType

A.1.3 Atributos dos elementos BPEL

A tabela 2 apresenta como os atributos BPEL são utilizados na notação ARIS. Cada atributo abaixo é utilizado por um conjunto de elementos BPEL. Os atributos que cada elemento possui estão descritos na seção A.1.4.

Tabela 2. Atributos BPEL utilizados na notação ARIS

Nome do atributo	Significado
Abstract process	Se esse atributo estiver habilitado, o processo associado é um cenário de negócio. Se o atributo não estiver habilitado, é um processo executável.
Enable instance compensation	Se esse atributo estiver habilitado, o processo como um todo pode ser revertido no caso de uma mudança realizada no barramento de serviço, como uma atualização do processo ou uma falha no barramento. Se o atributo estiver desabilitado, a instância do processo que estiver ativa durante o processo de manutenção do barramento é

Nome do atributo	Significado
	destruída sem compensação e as mudanças serão mantidas.
Expression language	Define uma linguagem XML usada para expressões. Se o atributo não for atualizado, o padrão usado será XPATH 1.0.
Join condition	Esse atributo contém uma expressão lógica (regra) de como links de entrada são conectados (padrão: OR).
Query language	Define uma linguagem XML usada para consultas. Se o atributo não for atualizado, o padrão usado será XPATH 1.0.
Suppress join failure	Se esse atributo estiver habilitado, tratamento de falhas estarão suspensos para junções.
Variable access serializable	Especifica a visibilidade e acesso para as variáveis locais dentro dos blocos (escopo).
Condition expression	Especifica as condições que devem ser conhecidas para um *switch* ser executado.
Create instance	Se esse atributo estiver habilitado, uma nova instância do processo é criada quando a atividade corrente é invocada.
Extension XML	Descreve os elementos da linguagem BPEL proprietária como extensões.
From expression	Contém uma expressão na linguagem de consulta que é usada como uma expressão de origem para associações da atividade Copy.
From literal	Contém uma expressão (numérica ou string) que é usada como uma expressão de origem para associações da atividade Copy.
Opaque	Ativa uma forma especial de associação, particular para processos abstratos (business protocol).
Type	Especifica se deve esperar até um tempo específico ou por um período específico. O atributo tem como valor as opções “Until” ou “For”.
Path condition	Especifica a condição que deve ser conhecida para aquele caminho ser processado.
Query expression	Descreve a consulta para identificar um valor dentro de uma variável de origem ou destino.

Nome do atributo	Significado
Role type	Especifica o papel do serviço correspondente. A definição de papéis está na especificação 1.1 do BPEL4WS. Um processo BPEL possui um serviço web como parceiro e pode ser parceiro de um serviço web (uma vez que o processo é acessado como um serviço descrito em WSDL). O valor do atributo Role Type poderá ser partner role, indicando um papel realizado por um parceiro, ou my role, indicando um papel realizado pelo próprio processo BPEL.
Default	Se esse atributo estiver marcado, o caso *default* é executado depois de um *switch,* se nenhum dos outros casos especificados for aplicado (vide Figura 100).
Sequence order	Descreve a posição da conexão na seqüência de conexões da atividade.
Transition condition	Especifica a condição para transações baseadas em link entre atividades.
Initiate	Um CorrelationSet já está iniciado se esse atributo estiver marcado.
Pattern	Define o padrão de mensagem (CorrelationSet) que será utilizado.

A.1.4Semântica da notação ARIS para os elementos BPEL

Nas próximas seções serão detalhadas as semânticas dos elementos da notação ARIS para BPEL. Cada seção possui um texto explicativo e uma figura explicitando a semântica do elemento.

A.1.4.1 Processo BPEL

Um processo BPEL e sua direção de execução são mapeados através do arranjo das atividades BPEL na seqüência correta e conectando-as umas às outras com um elemento BPEL ProcessStart como nó inicial e um elemento BPEL ProcessEnd como nó final.

Mais informações sobre a semântica podem ser encontradas na especificação 1.1 do BPEL4WS, seção 5.

Os seguintes atributos podem ser alterados para o elemento ProcessStart:

- Enable instance compensation
- Abstract Process
- Query Language

- Suppress Join Failure
- Expression Language

Os seguintes atributos podem ser alterados para uma atividade (básica ou estruturada).

- Join Condition
- Suppress Join Failure

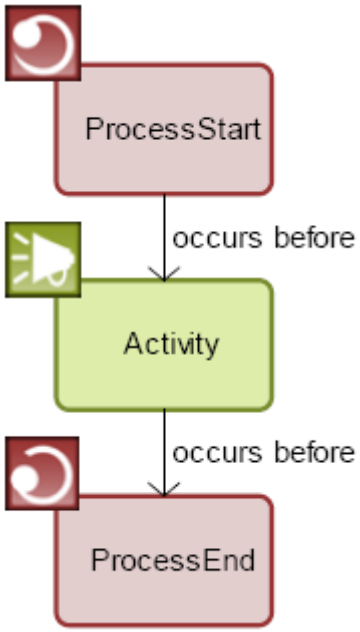


Figura 82 – Processo BPEL

A.1.4.2 Semântica de PartnerLinkType, PartnerLink, Partner e Target namespace

A especificação de um namespace ao qual o processo modelado pertence é mapeado pela conexão de um ProcessStart a um Namespace.

Um PartnerLink em um processo é definido pela sua conexão ao elemento ProcessStart. Cada PartnerLink é descrito por um PartnerLinkType. Um PartnerLinkType descreve os PortTypes usados para o relacionamento orientado a diálogo entre dois serviços e papéis que são alocados para os serviços nesse cenário. Um máximo de dois PartnerLinks diferentes podem ser descritos por um único PartnerLinkType. Por exemplo, fornecedores diferentes (e serviços) podem ser invocados em um processo de obtenção, mas todos eles podem ser caracterizados pelo mesmo PartnerLinkType. PartnerLink e PartnerLinkType são ligados por uma conexão de tipo **has type**. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 7.1.

A conexão de um PartnerLink a um ou mais PortTypes representa um uso corrente de PartnerLinkType desse PartnerLink. Nesse ambiente, para assegurar uma

modelagem mais realística de um processo de negócio, é extremamente importante que os relacionamentos requeridos com processos parceiros – incluindo os de fora da organização – possam ser representados. Um parceiro representa tanto um provedor de serviço como um consumidor de serviço. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 7.

Um parceiro em um processo é definido pela conexão de um ProcessStart a um Partner. Um Partner empacota um conjunto de PartnerLinks que definem o relacionamento entre dois processos parceiros.

Os seguintes atributos podem ser atualizados para relacionamento do tipo **links port types**:

- Role Type
- Connection Role (atributo padrão do ARIS para um relacionamento) – define um nome para um relacionamento entre serviços.

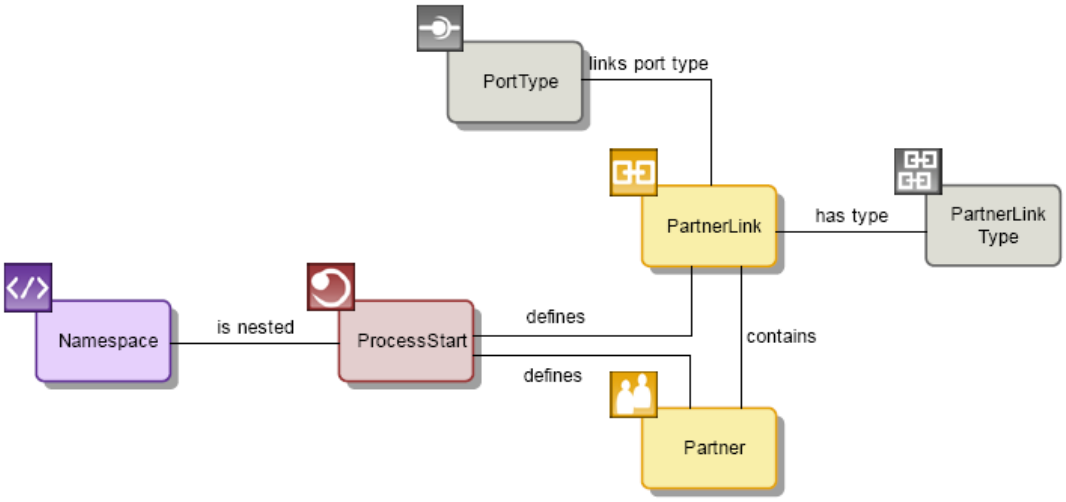


Figura 83 – PartnerLinkType, PartnerLink, Partner e Target namespace

A.1.4.3 Semântica de PortType, Operations, Parameters e seus MessageTypes

Para estender a descrição de um PortType WSDL para atingir os requerimentos da especificação do BPEL, conectamos o PortType a um ou mais Operations. Cada Operation define um ou mais Parameters e Faults. Parameters e Faults possuem um MessageType específico, o qual empacota um ou mais MessageParts. Um MessagePart pode tanto ser definido por um XSDType como por um XSDElement. Mais informações sobre a semântica pode ser encontrada na especificação 1.1 do W3C WSDL.

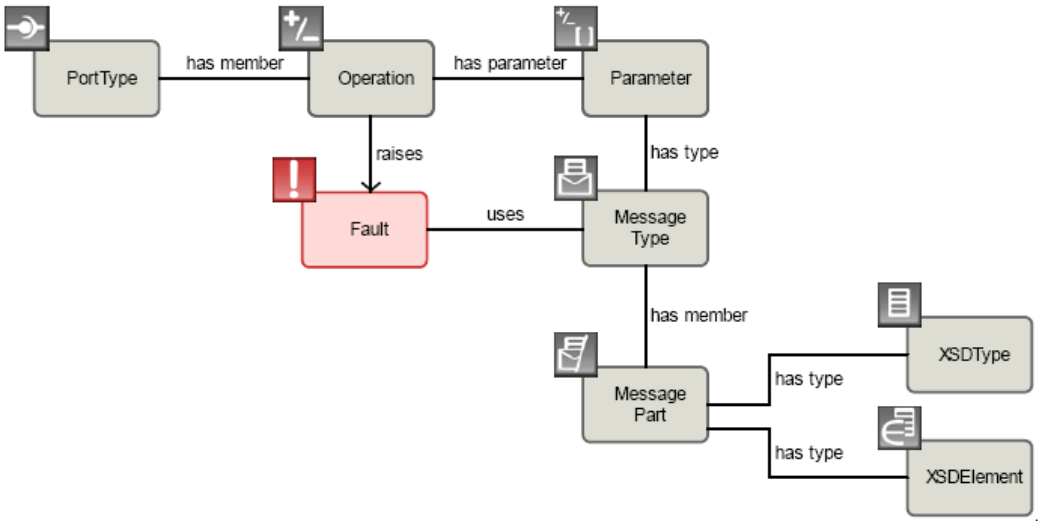


Figura 84 – PortType, Operations, Parameters e seus MessageTypes

A.1.4.4 Semântica dos elementos Variables

Processos de negócio modelam interações que possuem um estado. Esse estado consiste de mensagens que são enviadas ou recebidas, e também por outros dados importantes, tais como, valores de *timeout*. Para gerenciar o estado de um processo de negócio, são necessárias variáveis de estado, as quais são referenciadas pelos Variables no BPEL.

Uma variável no processo é definida pela conexão de um ProcessStart a um Variable. Em um escopo, uma variável é definida pela conexão de um Variable a um ScopeStart. Um tipo de variável é definido pela conexão de um XSDType, um XSDElement ou um MessageType a um Variable. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 9.

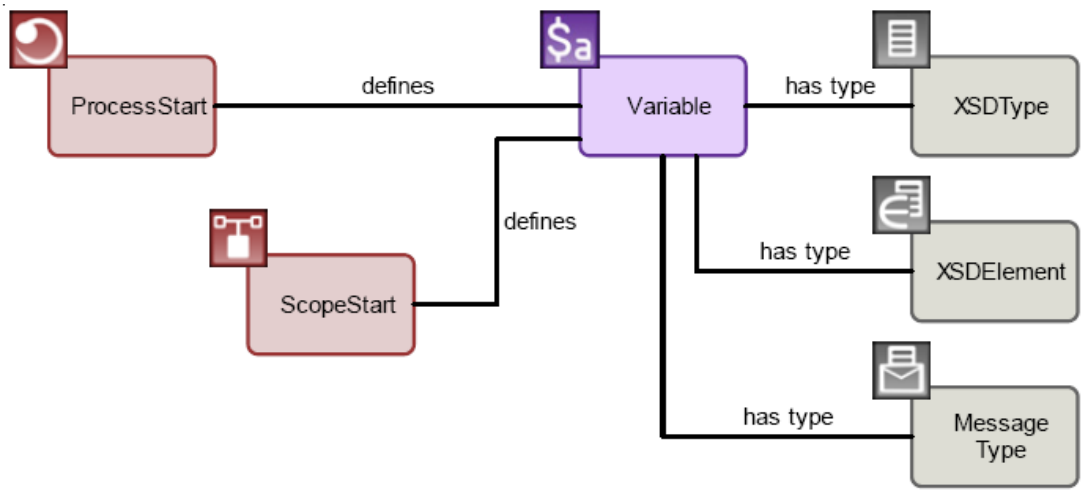


Figura 85 – Definição de Variable

A.1.4.5 Semântica de CorrelationSets e suas propriedades associadas

Um conjunto de correlação em um processo é definido pela conexão de um ProcessStart e um CorrelationSet. Um conjunto de correlação em um escopo é mapeado pela conexão de um ScopeStart e um CorrelationSet. Conectando-se um CorrelationSet a um ou mais Property definimos tanto o conteúdo de um conjunto de correlação quanto a propriedade em si (campo da mensagem). O tipo de propriedade é especificado pela conexão de um Property a um XSDType. Um nome pra propriedade é definido pela conexão de um Property a um MessageType e a um MessagePart. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seções 8 e 10.

Os seguintes atributos podem ser alterados em um relacionamento do tipo **is also known as**:

- Query Expression

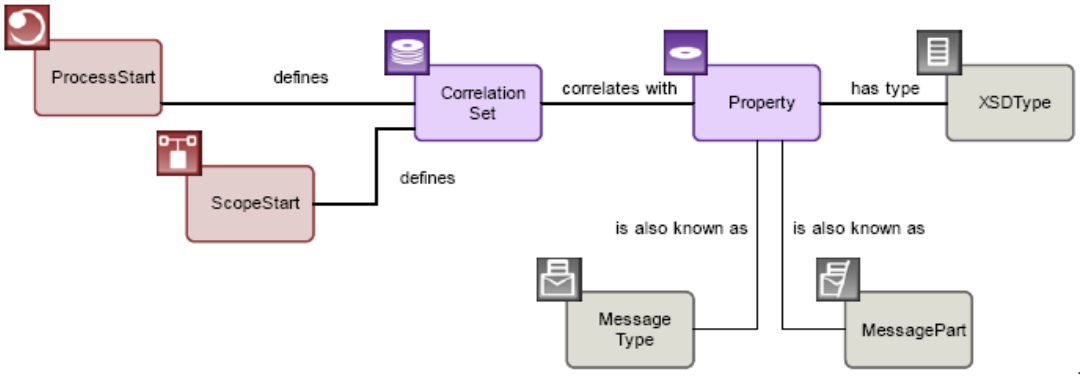


Figura 86 – CorrelationSet e propriedades relacionadas

A.1.4.6 Semântica do elemento Fault

Um tratamento de erro (modelado como um elemento Fault) em um processo é definido ao conectar um ProcessStart a um Catch. A parte do tratamento de erro que captura todos os erros no processo é especificado ao conectar um ProcessStart a uma atividade básica ou estruturada. Um tratamento de erro em um escopo é definido ao conectar um ScopeStart a um Catch. Um tratamento de erro que apenas captura certos erros em um processo ou escopo é definido ao conectar um Catch a um Fault especificado. Um Variable é gerado como resultado e uma atividade básica ou estruturada é executada. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 13.4.

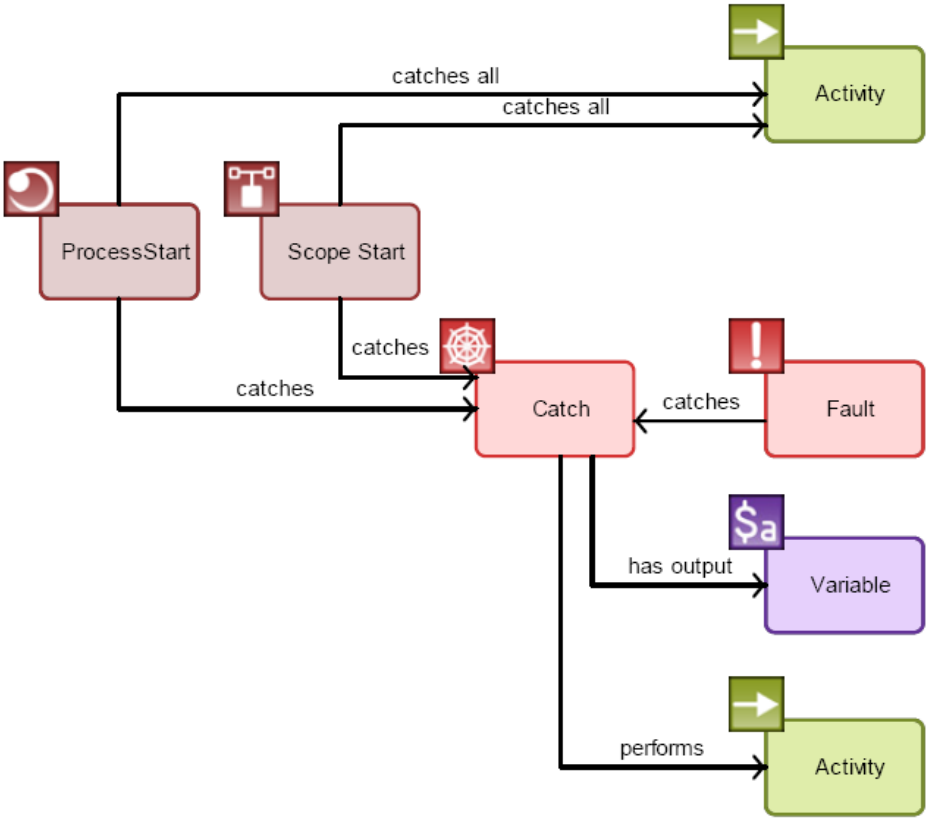


Figura 87 – Definição do tratamento Fault

A.1.4.7 Semântica do elemento Event

Um tratamento de evento (modelado como um elemento Event) em um processo é especificado pela conexão de um ProcessStart a um OnAlarm, a um OnMessage ou ambos. Um tratamento Event em um escopo é definido pela conexão de um ScopeStart a um OnAlarm, a um OnMessage, ou a ambos. Um tratamento de evento que lida com eventos de alarme em um processo ou escopo é definido pela conexão de um OnAlarm a uma atividade básica ou estruturada. Um tratamento de evento que trata eventos de mensagens dentro de um processo ou escopo é definido pela conexão de um OnMessage a um PartnerLink, a um Operation que envia uma mensagem, à variável gerada, ao CorrelationSet em uso e à uma atividade básica ou estruturada que é executada na mensagem. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 13.5.

Os seguintes atributos podem ser alterados usando OnAlarm:

- Type
- Condition Expression

Os seguintes atributos podem ser alterados usando um relacionamento do tipo **uses**:

- Initiate

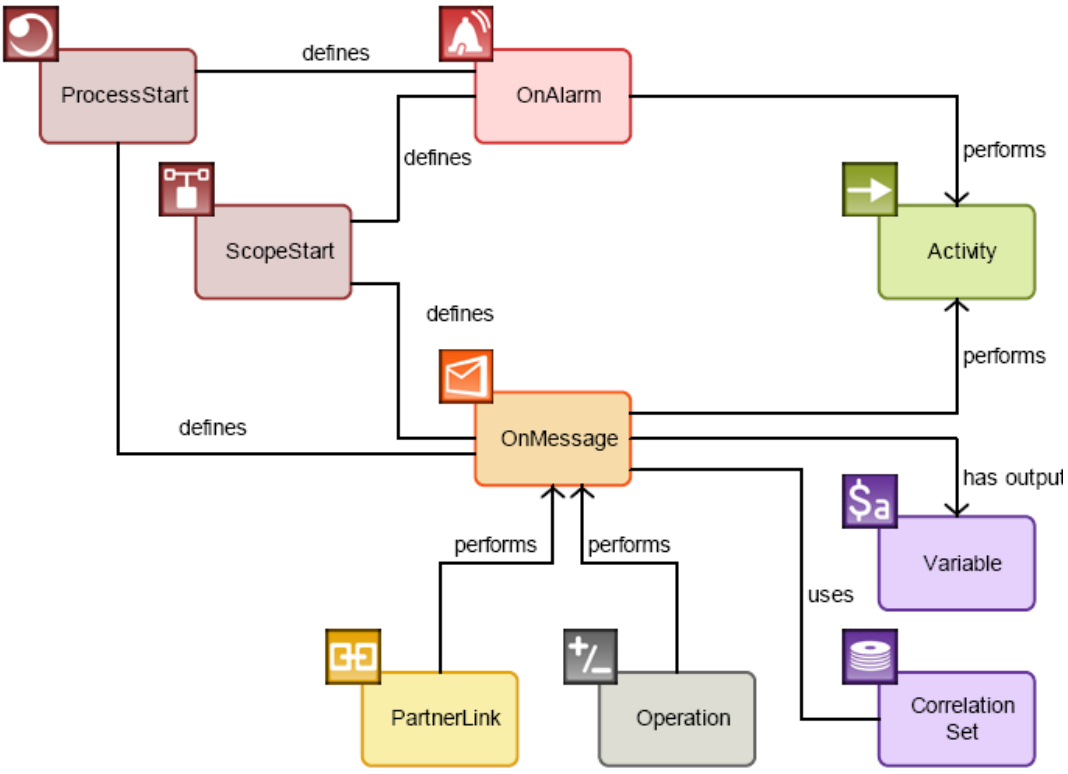


Figura 88 – Definição do tratamento Event

A.1.4.8 Semântica do tratamento de compensação

Um tratamento de compensação em um processo é especificado pela conexão de um ProcessStart a uma atividade básica ou estruturada usando um tipo de conexão específica do ARIS. Um tratamento de compensação em um escopo é definido pela conexão de um ScopeStart a uma atividade básica ou estruturada usando um tipo de conexão específico do ARIS, chamado de *defines compensation*. Assim, a atividade ligada como *defines compensation* a um processo ou a um escopo será executada quando um pedido de compensação for emitido (por exemplo, ao executar um elemento Compensate). Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 13.3.

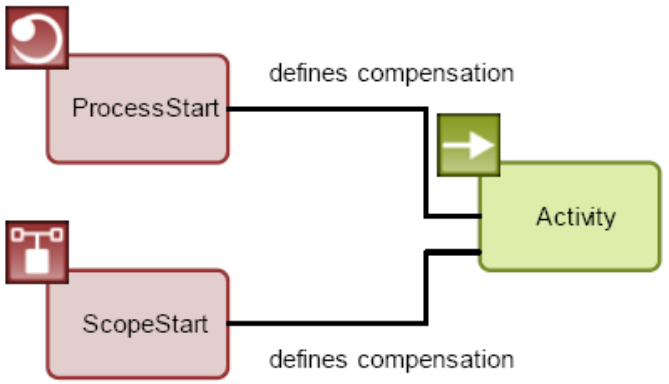


Figura 89 – Definição do tratamento de compensação

A.1.4.9 Semântica do elemento Scope e uso da atividade Compensation

Um escopo (modelado como um elemento Scope) é uma atividade BPEL que começa com ScopeStart e termina com ScopeEnd. Um Scope compensado é definido pela conexão de um Compensate a seu ScopeStart e é compensado com o uso da atividade básica Compensate. Um Invoke compensado é definido pela conexão de um Compensate a um Invoke e é compensado com o uso da atividade básica Compensate. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 13.3.

O seguinte atributo pode ser atualizado para o ScopeStart:

- Variable Access Serializable

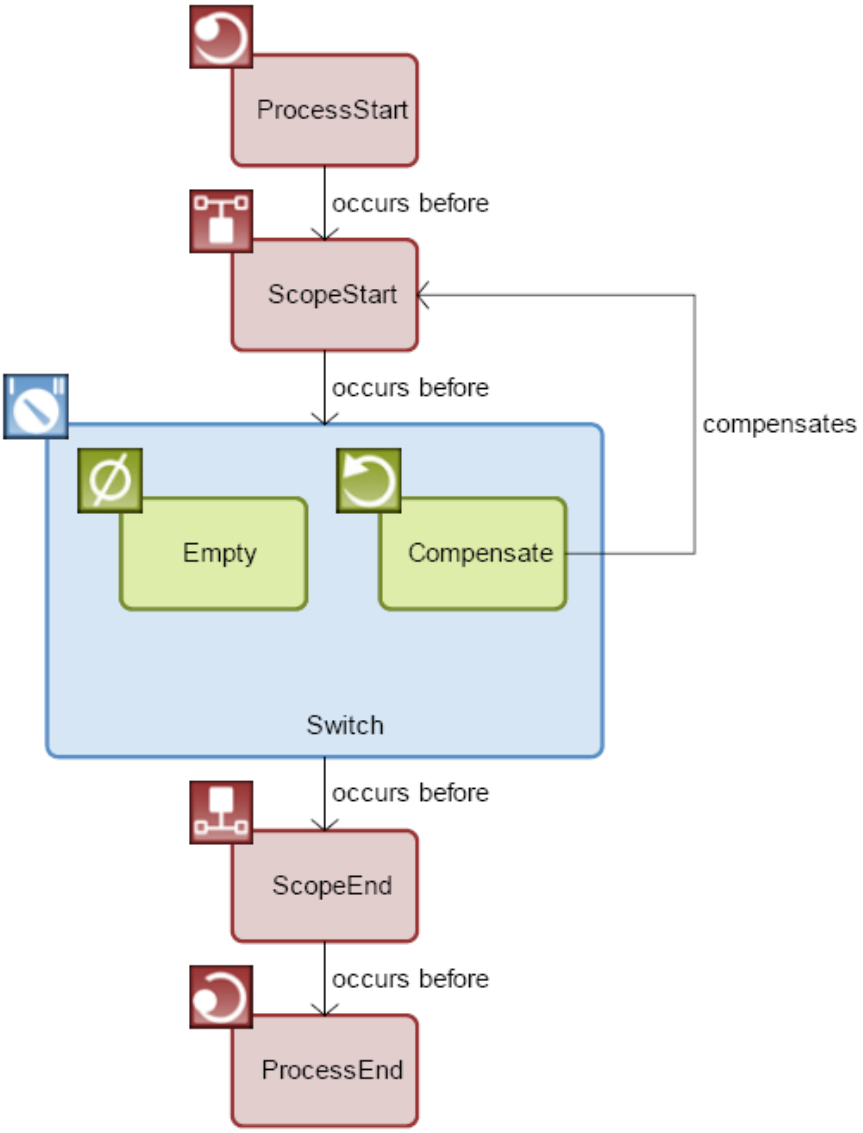


Figura 90 – Definição do Scope e uso da atividade Compensation

A.1.4.10 Usando a atividade Invoke

Um PartnerLink e um Operation que estão conectados a uma atividade básica Invoke definem o serviço invocado. Para definir uma variável de entrada, um Variable é conectado a uma conexão de entrada ao Invoke. Para definir uma variável de saída, um Variable é conectado a uma conexão de saída ao Invoke. Uma conexão entre CorrelationSet e um Invoke definem o conjunto de correlação usado. Dependendo do tipo de conexão do ARIS que conecta um Invoke a uma atividade básica ou estruturada, ou uma parte de um tratamento Fault ou um tratamento Compensation é definido. A conexão entre um Invoke e um Catch define outra parte de um tratamento Fault. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 11.3.

Os seguintes atributos podem ser atualizados para o relacionamento do tipo **use**:

- Initiate
- Pattern

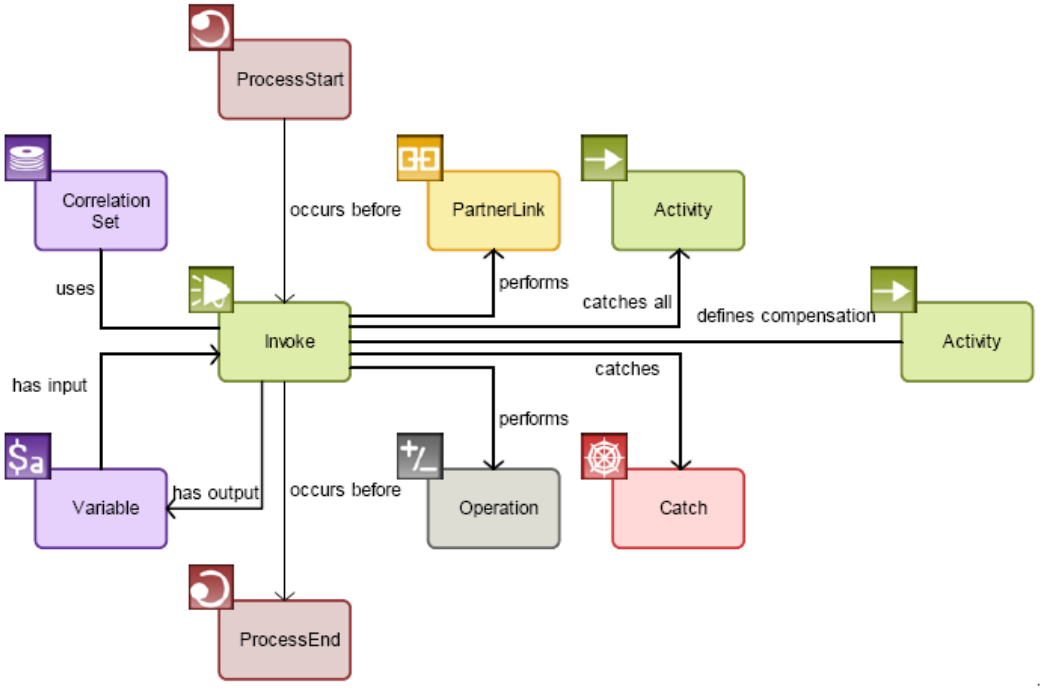


Figura 91 – Usando Invoke

A.1.4.11 Usando a atividade Receive

Um PartnerLink e um Operation que estão conectados a uma atividade básica Receive definem o serviço invocado. Para definir uma variável de saída, um Variable é conectado a uma conexão de saída ao Receive. Uma conexão entre CorrelationSet e um Receive definem o conjunto de correlação usado. Apenas uma atividade Receive deve ser usada para um específico PartnerLink, PortType, Operation e CorrelationSet numa instância do processo de negócio. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 11.4.

O seguinte atributo pode ser atualizado:

- Create Instance

O seguinte atributo pode ser atualizado para o relacionamento do tipo **use**:

- Initiate

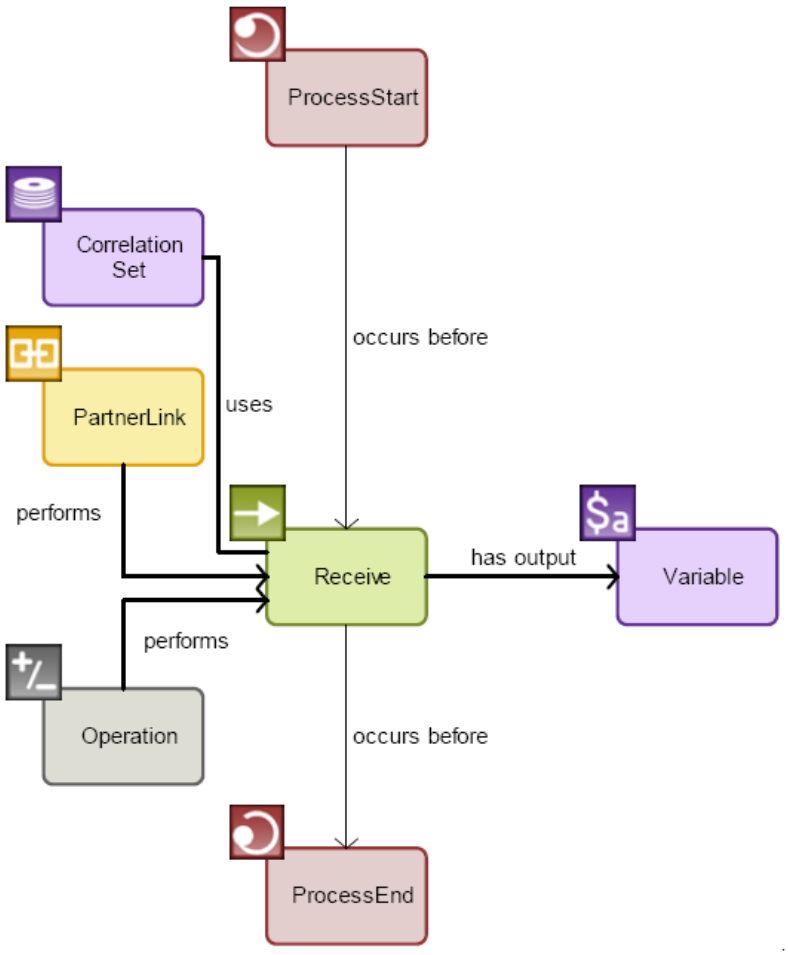


Figura 92 – Usando Receive

A.1.4.12 Usando a atividade Reply

Um PartnerLink e um Operation que estão conectados a uma atividade básica Reply definem o serviço invocado. Para definir uma variável de entrada, um Variable é conectado a uma conexão de entrada ao Reply. Uma conexão entre CorrelationSet e um Reply definem o conjunto de correlação usado. A conexão entre Reply e Fault especifica um padrão que é lançado por uma atividade. Apenas uma atividade Reply deve ser usada para um específico PartnerLink, PortType, Operation e CorrelationSet numa instância do processo de negócio. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 11.4.

O seguinte atributo pode ser atualizado para o relacionamento do tipo **use**:

- Initiate

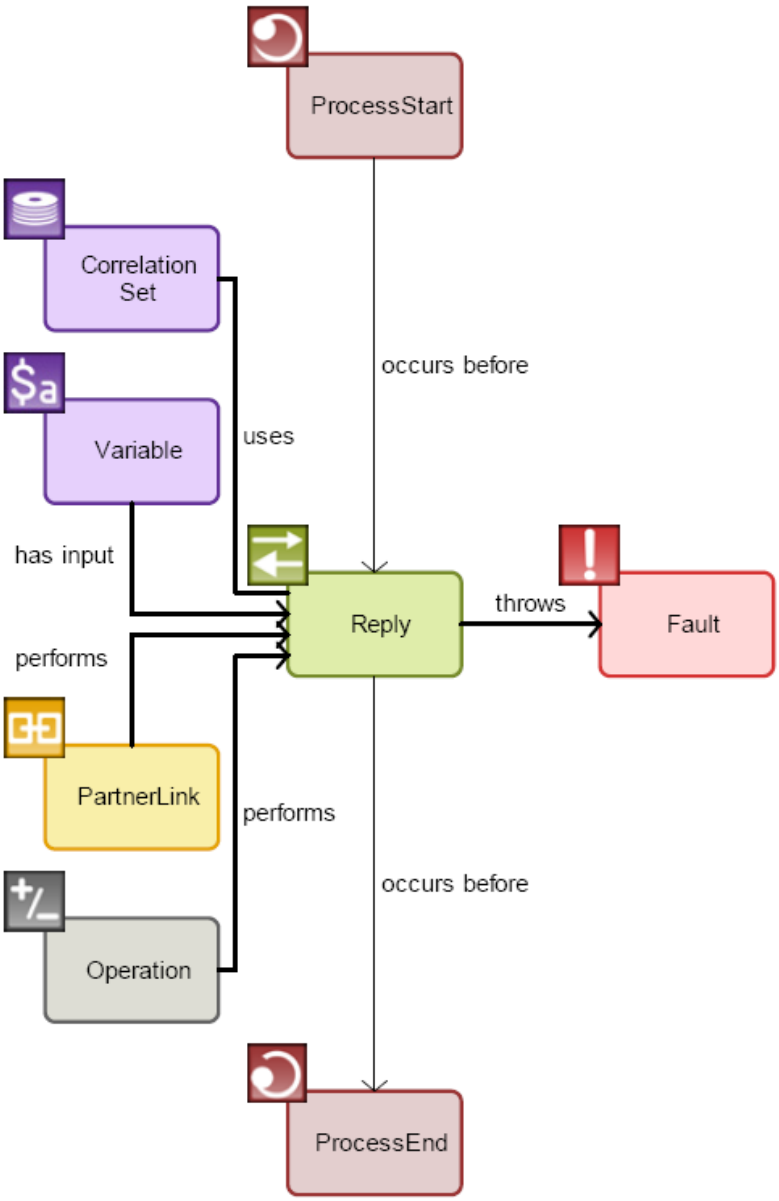


Figura 93 – Usando Reply

A.1.4.13 Usando a atividade Assign

Construções de cópia para a atividade básica Assign são definidas pela conexão de um Assign a um ou mais Copy. O conteúdo de um Copy é especificado pela sua conexão a elementos de entrada e saída. Eles podem ser tanto PartnerLink como Variable com MessagePart se a variável for do tipo Message ou uma Variable com um Property se existir uma nome de propriedade. A distinção entre elementos de entrada e saída é percebida pela direção das conexões com Copy. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 11.5.

Os seguintes atributos podem ser atualizados pro Copy:

- From expression
- From Literal

- Opaque

Os seguintes atributos podem ser atualizados para o relacionamento do tipo **perform**:

- Sequence Order

Os seguintes atributos podem ser atualizados para o relacionamento do tipo **from** entre Copy e Variable:

- Query Expression

Os seguintes atributos podem ser atualizados para o relacionamento do tipo **to** entre Copy e Variable:

- Query Expression

Os seguintes atributos podem ser atualizados para o relacionamento do tipo **from** entre Copy e PartnerLink:

- Role Type

Os seguintes atributos podem ser atualizados para o relacionamento do tipo **to** entre Copy e PartnerLink:

- Role Type

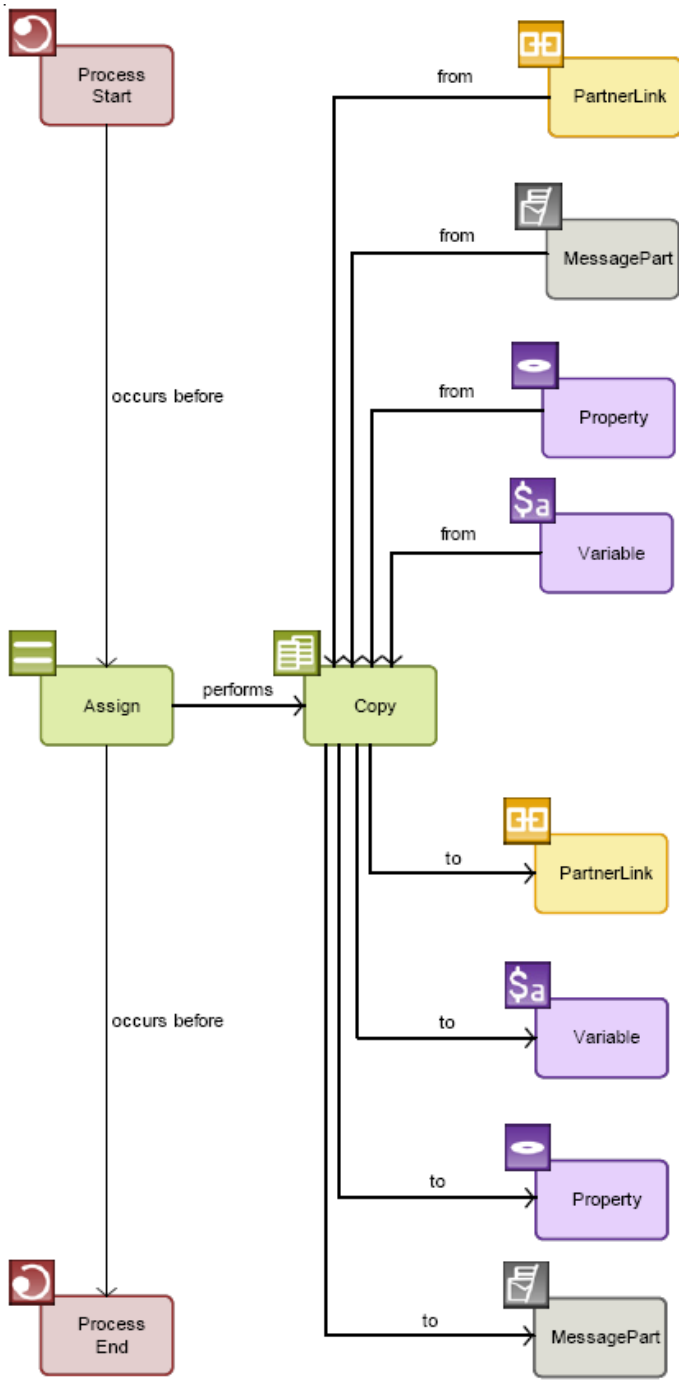


Figura 94 – Usando Assign

A.1.4.14 Usando a atividade Throw

A conexão entre Variable e Throw usando uma conexão de entrada a Throw define a variável de entrada usada pela atividade básica Throw. A conexão entre Throw e Fault especifica o padrão que é lançado por uma atividade. Todo Fault deve ter um nome único. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 11.5.

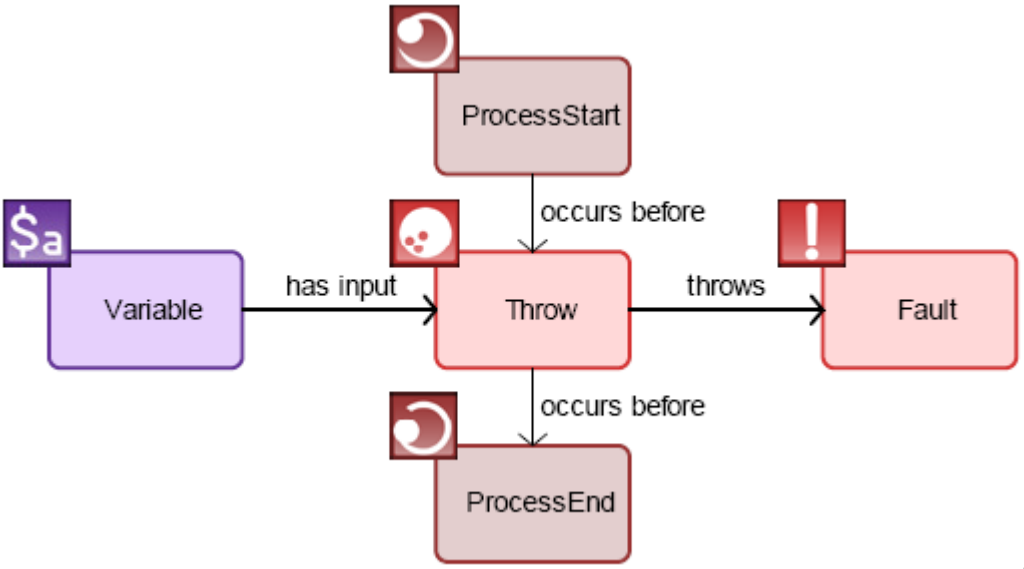


Figura 95 – Usando Throw

A.1.4.15 Usando a atividade Empty

Empty é requerido, por exemplo, se padrões forem capturados e suprimidos, e permite que uma instrução no-op seja usada no processo de negócio. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 11.8.

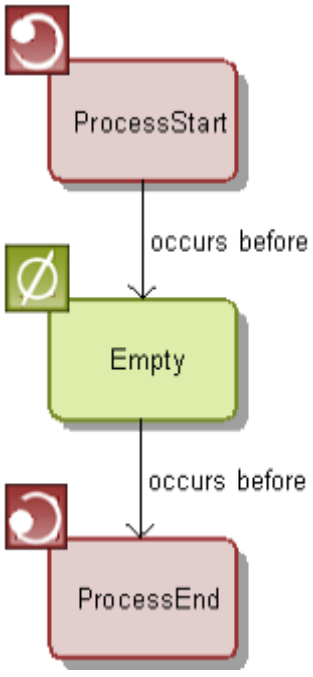


Figura 96 – Usando Empty

A.1.4.16 Usando a atividade Wait

Wait é usado em um processo de negócio para espera tanto por um período de tempo especificado como até um determinado tempo. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 11.7.

Os seguintes atributos podem ser atualizados para Wait:

- Type
- Condition Expression

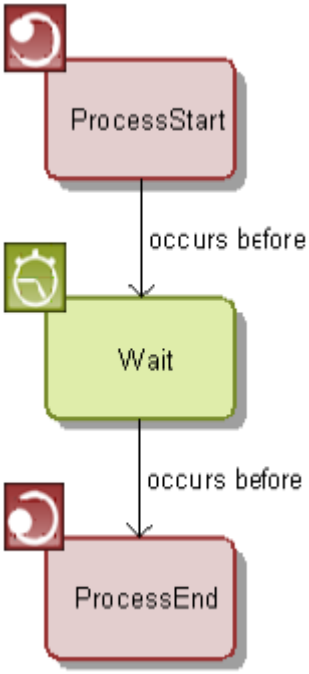


Figura 97 – Usando Wait

A.1.4.17 Usando a atividade Terminate

Terminate é usado para terminar uma instância de um processo de negócio imediatamente. Todas as atividades rodando naquele momento são canceladas o mais rápido possível sem tratamento Fault ou com compensação (execução da atividade Compensation). Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 14.6.

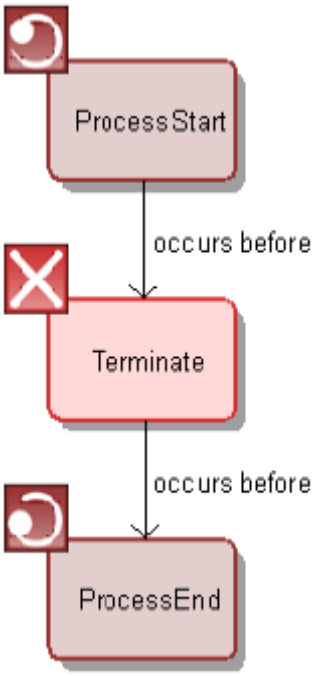


Figura 98 – Usando Terminate

A.1.4.18 Seqüência implícita e o uso da atividade Sequence

A atividade estruturada Sequence pode ser representada de duas maneiras: (a) implicitamente, como uma série de atividades conectadas umas nas outras; (b) como um Sequence contendo atividades. No último caso, as atividades aninhadas devem ser conectadas ao Sequence usando uma conexão implícita do tipo **stars with**. As atividades são executadas na ordem em que são colocadas. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 12.1.

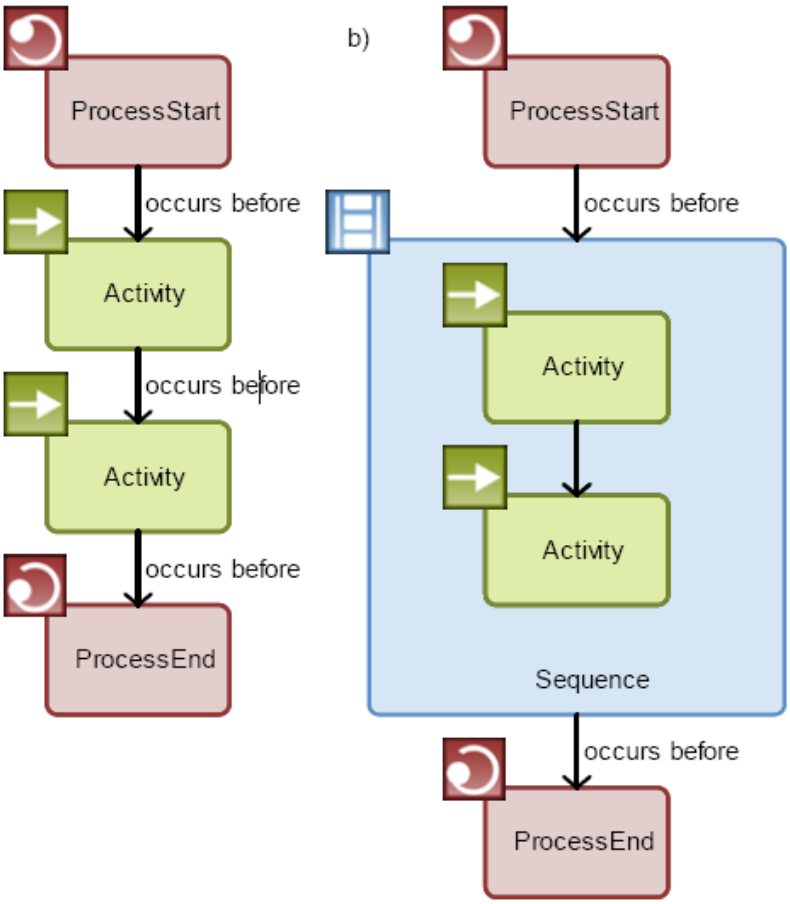


Figura 99 – Seqüência implícita e o uso da atividade Sequence

A.1.4.19 Usando a atividade Switch

O conteúdo de uma atividade estruturada Switch é definido pela colocação de atividades dentro de Switch, as quais são conectadas ao Switch por uma conexão implícita do tipo **has case**. As atividades básicas dentro do ramo do Switch são processadas pela ordem determinada pelo ramo. Se nenhuma das condições determinadas ocorrer, um ramo **default** é processado. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 12.2.

Os seguintes atributos podem ser atualizados para o relacionamento do tipo **has case**:

- Condition Expression
- Sequence Order
- Default

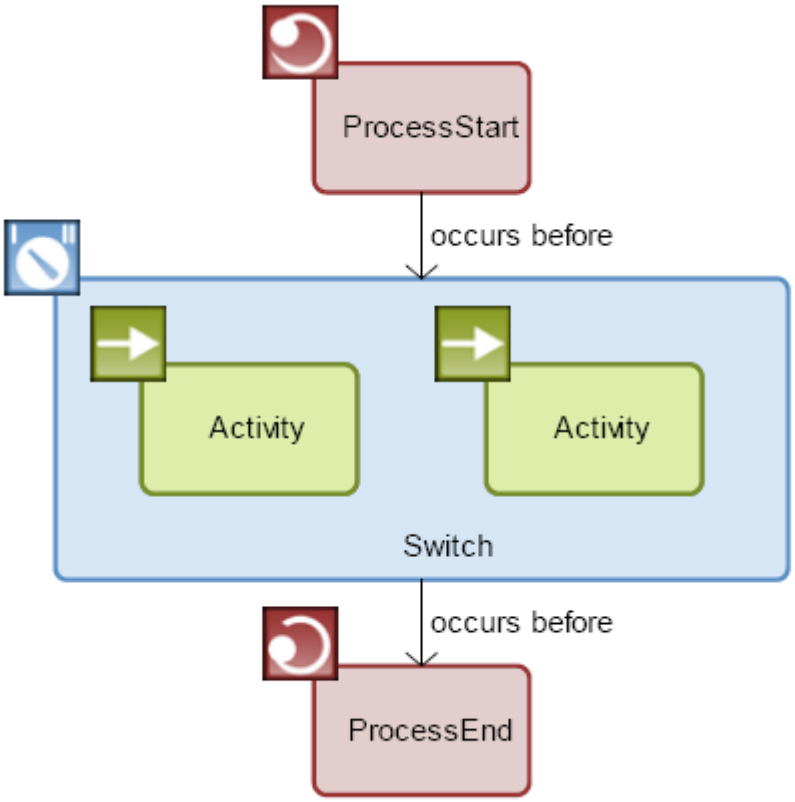


Figura 100 – Usando Switch

A.1.4.20 Usando a atividade While

O conteúdo de uma atividade estruturada While é definido pela colocação de atividades dentro do While, as quais são conectadas ao While por uma conexão implícita do tipo **perform**. A interação é executada enquanto uma determinada condição ocorre. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 12.3.

Os seguintes atributos podem ser atualizados para o While:

- Path Condition

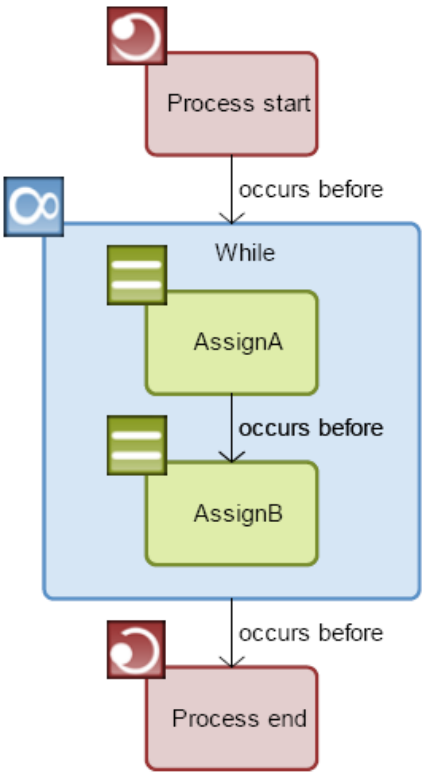


Figura 101 – Usando While

A.1.4.21 Usando a atividade Pick

O conteúdo de uma atividade estruturada Pick é definido pela conexão de Pick a um OnAlarm ou a um OnMessage ou aos dois. Pick espera por um ou mais eventos e então executa a atividade que está ligada ao evento. Se mais de um evento permitido ocorrer, a atividade é executada para o primeiro evento que ocorrer. Uma vez tendo aceitado o evento, todos os outros eventos que seriam normalmente aceitos, são rejeitados. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 12.4.

Os seguintes atributos podem ser atualizados para o Pick:

- Create Instance

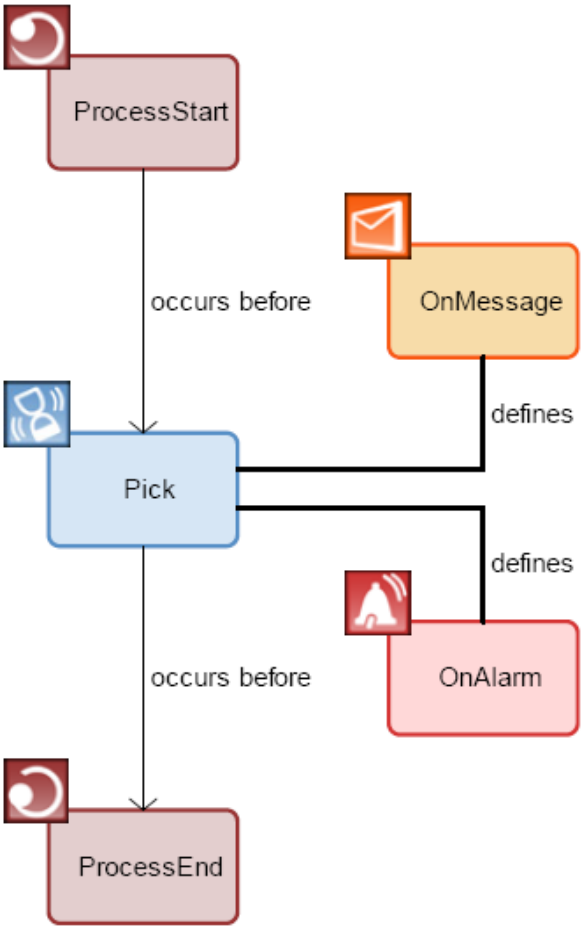


Figura 102 – Usando Pick

A.1.4.22 Usando a atividade Flow

O conteúdo de uma atividade estruturada Flow é definido pela colocação de atividades dentro do Flow, as quais são conectadas ao Flow por uma conexão implícita do tipo **perform**. Flow cria um conjunto de atividades que executa simultaneamente. Links sincronizam as atividades dentro de Flow e é representado por linhas pontilhadas. Mais informações sobre a semântica pode ser encontrada em na especificação 1.1 do BPEL4WS, seção 12.5.

Os seguintes atributos podem ser atualizados para o relacionamento do tipo l**inks**:

- Transition Condition

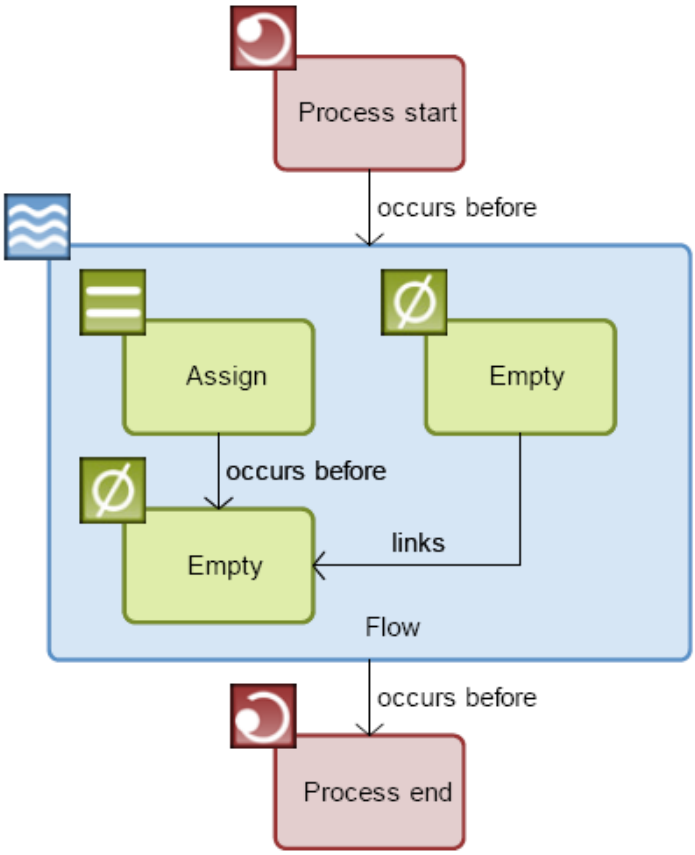


Figura 103 – Usando Flow

A.1.4.23 Usando a atividade Extension

Um PartnerLink e um Operation que estão conectados a uma atividade ARIS BPEL Extension definem um serviço que invocam uma atividade. A conexão entre um Variable e um Extension usando uma conexão de saída do Extension define a variável de saída usada. A conexão entre CorrelationSet e Extension define o conjunto de correlação usado.

Os seguintes atributos podem ser atualizados para o relacionamento do tipo **uses**:

- Initiate

Os seguintes atributos podem ser atualizados para o Extension:

- Extension XML

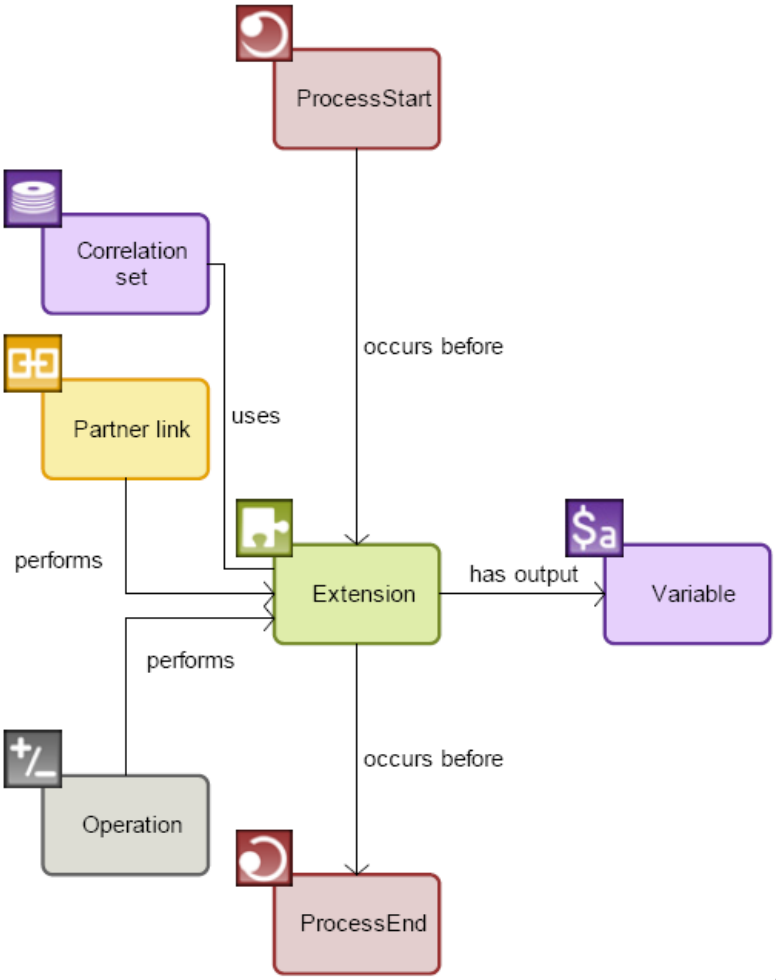


Figura 104 – Usando Extension -1

Um PartnerLink e um Operation que estão conectados a uma atividade ARIS BPEL Extension definem um serviço invocado. A conexão entre um Variable e um Extension usando uma conexão de entrada ao Extension definem a variável de entrada usada. A conexão entre CorrelationSet e Extension define o conjunto de correlação usado.

Os seguintes atributos podem ser atualizados para o relacionamento do tipo **uses**:

- Initiate

Os seguintes atributos podem ser atualizados para o Extension:

- Extension XML

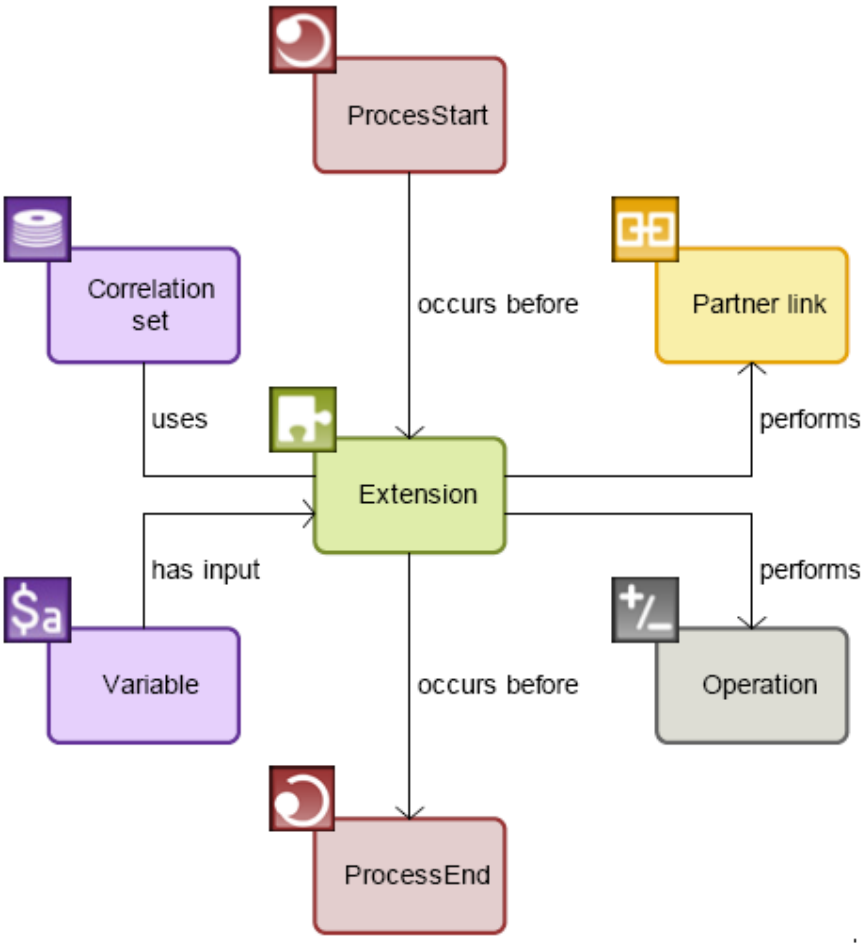


Figura 105 – Usando Extension -2

A.1.5Extensibilidade BPEL na notação ARIS BPEL

O objetivo do padrão BPEL é definir descrições de processos que não estão diretamente ligados a uma plataforma de implementação específica. Uma ferramenta, para visualizar a definição do processo BPEL, deve então prover uma opção para especificar uma dada extensão específica para o padrão BPEL, porém uma forma genérica.

A atividade Extension ARIS BPEL permite que extensões BPEL que são específicas a uma determinada plataforma sejam incluídas em um processo lógico. Dessa forma, não há quebras no processo lógico se uma extensão BPEL de execução em uma plataforma específica for adicionada ao processo. Além disso, modeladores BPEL podem gerar uma definição de processo BPEL válido através da adição de código XML na extensão BPEL como uma propriedade da atividade BPEL ARIS Extension. Ao mesmo tempo, a definição da atividade Extension como uma atividade regular permite a visualização gráfica de sua semântica, tais como a chamada a um *web service* ou escrever em uma variável de saída.

A.1.5.1 Atividade Extension chamada por um serviço

A Figura 106 abaixo representa o primeiro cenário de utilização da atividade Extension, no qual um PartnerLink e um Operation, os quais estão conectados à atividade por conexões de saída, definem um serviço que invoca a atividade. A conexão de Variable com Extension por uma conexão de saída do Extension define a variável de saída usada. A conexão do CorrelationSet ao Extension define o conjunto de correlação usado.

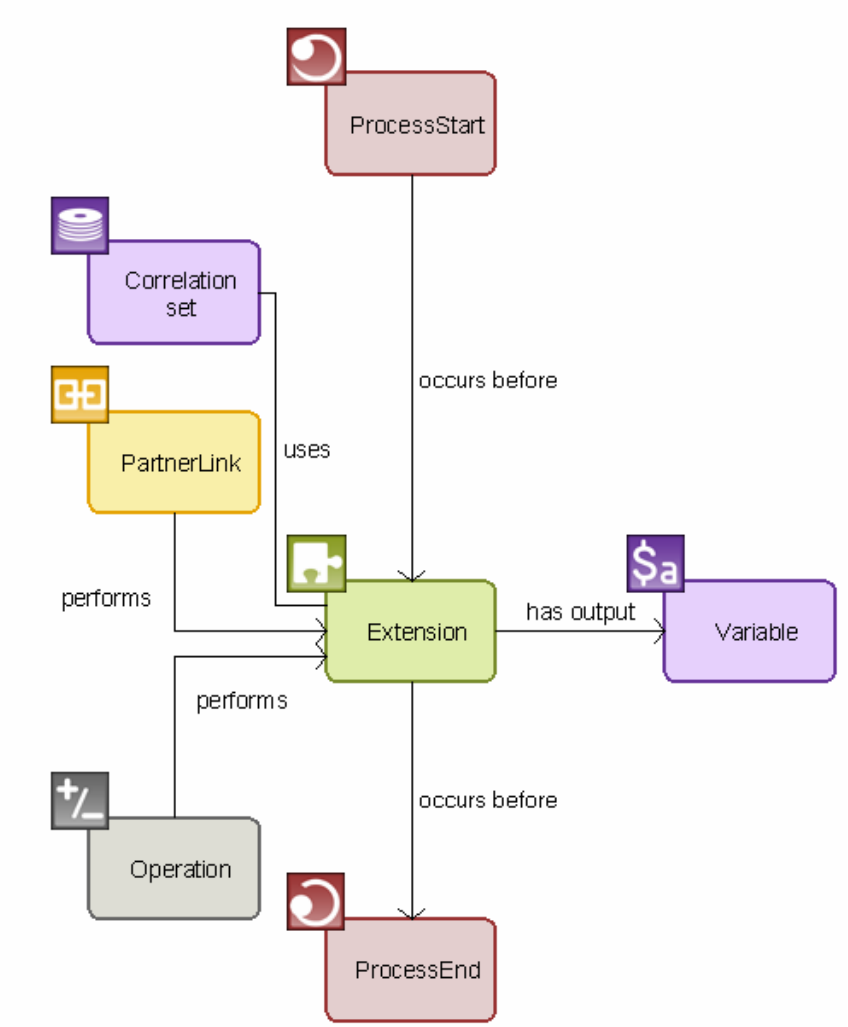


Figura 106 – Atividade Extension chamada por um serviço

O XML da Figura 107 abaixo corresponde ao processo modelado na Figura 106 acima. Ele assume que PartnerLink foi definido corretamente e se conecta a um tipo de porta PortType, a qual é definida no namespace http://temp.uri/. Note que o código XML abaixo não é executável porque a tag XML "bpelx", a qual é gerada pela atividade Extension, não é definida na especificação BPEL.

O código XML pode ser transformado em um processo BPEL executável pela execução de um passo a mais depois da geração. Nesse caso, a tag "bpelx" é substituída por uma tag de execução de um de plataforma específica, por exemplo, bpelx: exec xmlns:bpex= 'http://schemas.oracle.com/bpel/extension.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by the ARIS Business Architect, IDS Scheer AG. All
rights reserved. www.ids-scheer.com-->
<process expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116" name="ProcessStart"
queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
targetNamespace="http://www.ids-scheer.com/bpel"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:tns="http://www.ids-scheer.com/bpel/wsdl">
<bpelx name="Extension" operation="Operation"
outputVariable="Variable" partnerLink="PartnerLink"
portType="imp:portType" xmlns:imp="http://temp.uri/">
<correlations>
<correlation set="Correlation_x20_set"/>
</correlations>
</bpelx>
</process>
```

Figura 107 – Código XML de um processo BPEL -1

A.1.5.2 Atividade Extension chamando um serviço

A Figura 108 abaixo representa o segundo cenário de utilização da atividade Extension, no qual um serviço invocado é definido por um PartnerLink e um Operation, os quais estão conectados à atividade por conexões de entrada. A conexão de Variable com Extension por uma conexão de entrada do Extension define a variável de entrada usada. A conexão do CorrelationSet ao Extension define o conjunto de correlação usado.

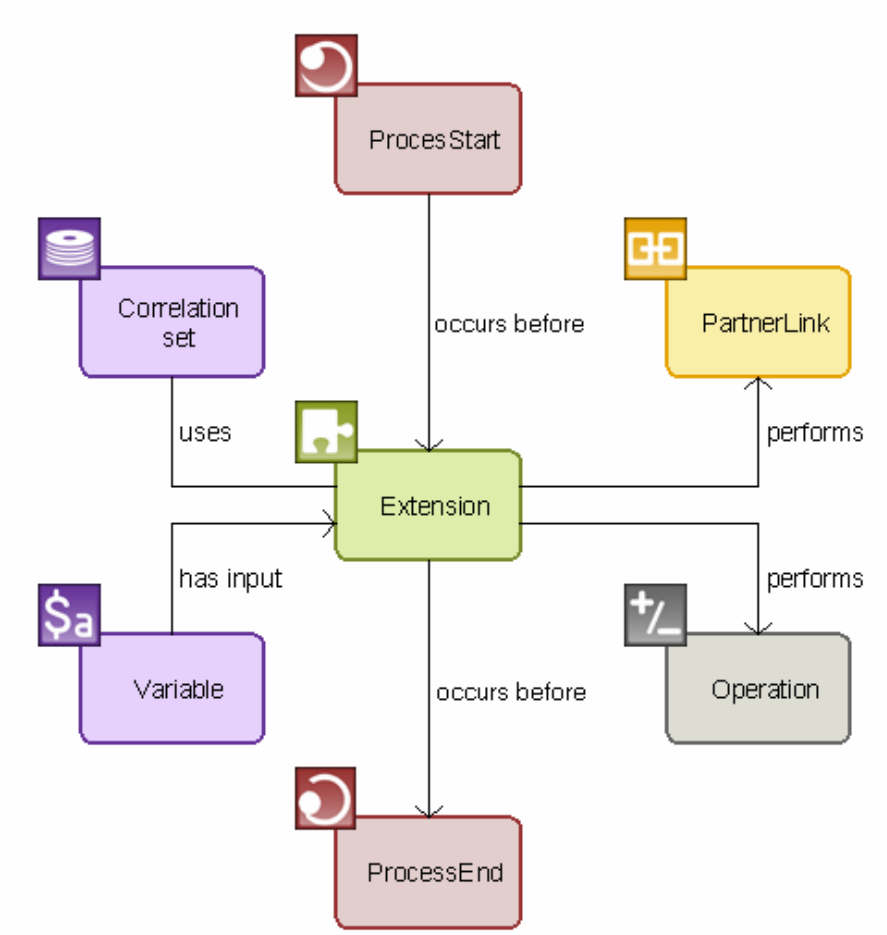


Figura 108 – Atividade Extension chamando um serviço

O XML da Figura 109 corresponde ao processo modelado na Figura 108. Ele assume que PartnerLink foi definido corretamente e se conecta a um tipo de porta PortType, a qual é definida no namespace http://temp.uri/. Note que o código XML abaixo não é executável porque a tag XML "bpelx", a qual é gerada pela atividade Extension, não é definida na especificação BPEL.

O código XML pode ser transformado em um processo BPEL executável pela execução de um passo a mais depois da geração. Nesse caso, a tag "bpelx" é substituída por uma tag de execução de um de plataforma específica, por exemplo, bpelx: exec xmlns:bpex= 'http://schemas.oracle.com/bpel/extension.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by the ARIS Business Architect, IDS Scheer AG. All
rights reserved. www.ids-scheer.com-->
<process expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116" name="ProcesStart"
queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
targetNamespace="http://www.ids-scheer.com/bpel"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:tns="http://www.ids-scheer.com/bpel/wsdl">
<bpelx inputVariable="Variable" name="Extension"
operation="Operation" partnerLink="PartnerLink"
portType="imp:portType" xmlns:imp="http://temp.uri/">
<correlations>
<correlation set="Correlation_x20_set"/>
</correlations>
</bpelx>
</process>
```

Figura 109 – Código XML de um processo BPEL -2

A.1.5.3 Atividade Extension contendo código XML

A Figura 110 representa um processo com código de execução Java na plataforma de execução Oracle. O seguinte código de extensão Oracle é mantido no atributo BPEL ExtensionXML da atividade Extension:

```
<bpelx:exec xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
language="java" version="1.4" ><![CDATA[String my = new String();]]>
</bpelx:exec>
```

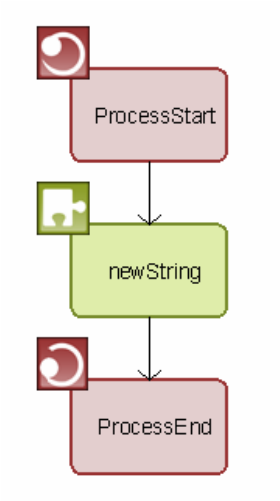


Figura 110 – Atividade Extension contendo código XML

O XML da Figura 111 corresponde ao processo modelado na Figura 110. Note que o código XML abaixo só pode ser executado na plataforma de execução ORACLE, pois propriedades de extensão do Oracle foram utilizadas.

```
<?xml version='1.0' encoding='UTF-8'?>
<!--Generated by the ARIS Business Architect, IDS Scheer AG. All
rights reserved. www.ids-scheer.com-->
<process expressionLanguage='http://www.w3.org/TR/1999/REC-xpath-
19991116' name='ProcessStart'
queryLanguage='http://www.w3.org/TR/1999/REC-xpath-19991116'
targetNamespace='http://www.ids-scheer.com/bpel'
xmlns='http://schemas.xmlsoap.org/ws/2003/03/business-process/'
xmlns:tns='http://www.ids-scheer.com/bpel/wsdl'>
<bpelx:exec language='java' name='newString' version='1.4'
xmlns:bpelx='http://schemas.oracle.com/bpel/extension'>
<![CDATA[String my = new String();]]> </bpelx:exec>
</process>
```

Figura 111 – Código XML de um processo BPEL em plataforma Oracle

A Figura 112 apresenta a tela do ARIS para inserção de código XML de extensão como valor do atributo ExtensionXML.

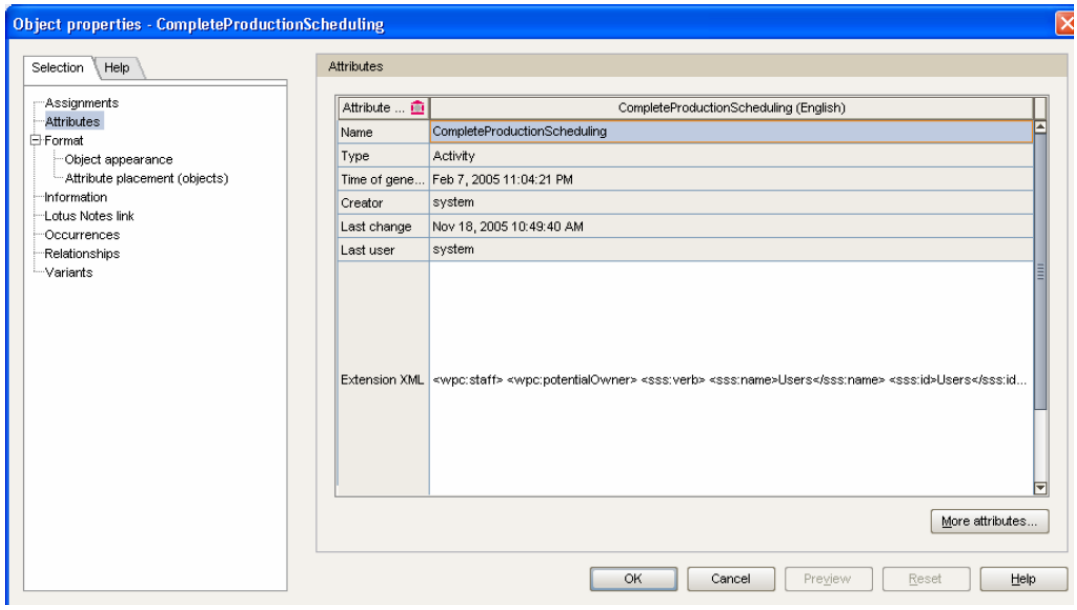


Figura 112 – Um exemplo do atributo ExtensionXML sendo utilizado

A.1.5.4 Atividade Extension como uma extensão da atividade estruturada BPEL

A atividade Extension pode ser usada tanto como uma atividade básica como uma atividade estruturada. As atividades estruturadas se diferenciam das básicas pelo fato de poderem conter outras atividades BPEL (tanto estruturadas como básicas). No cenário mencionado na seção acima, são é apresentada a atividade Extension como atividade básica. Porém, existem plataformas de execução de processo que podem também estender atividades estruturadas.

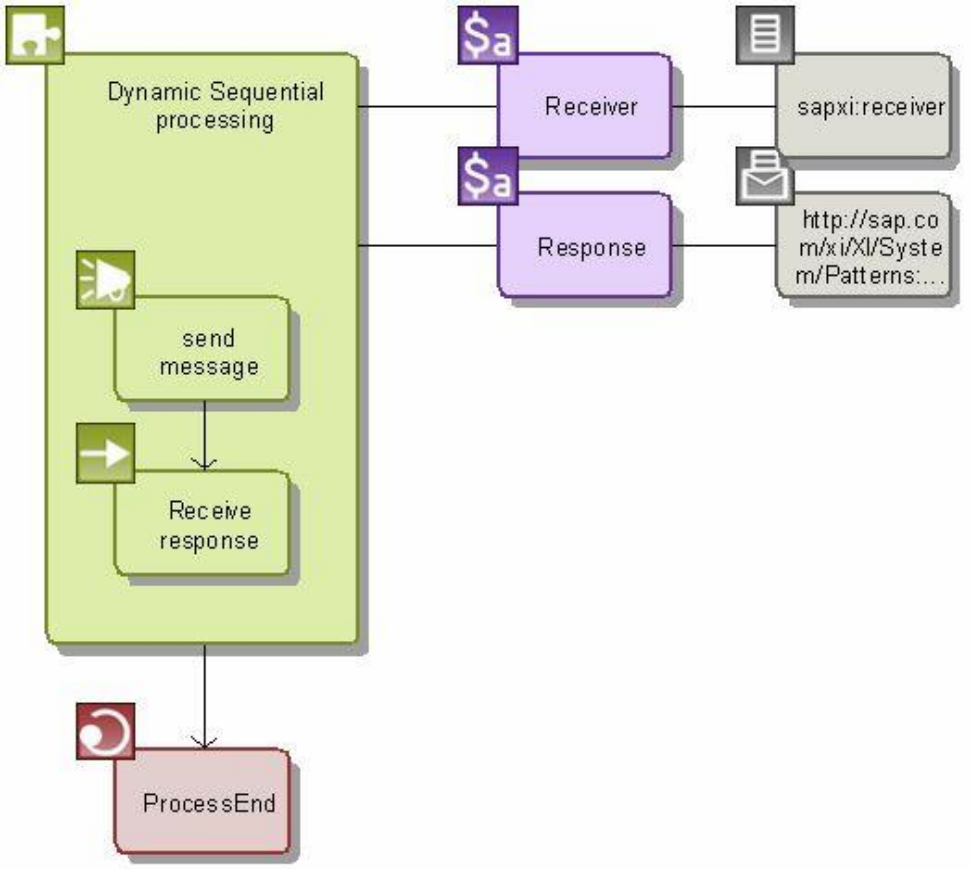


Figura 113 – Atividade Extension como atividade estruturada

A Figura 113 acima mostra a utilização da atividade Extension como atividade estruturada, o que se assemelha com o padrão "Scope". Conectando-se um Variable a um Extension via relacionamento do tipo **defines**, é definida uma variável local. Conectando-se qualquer atividade BPEL a um Extension usando o relacionamento do tipo **performs**, é definida uma atividade aninhada.

O XML da Figura 114 abaixo corresponde ao processo modelado na Figura 113 acima. Note que o código XML abaixo pode ser executado na plataforma de execução SAP, pois propriedades de extensão do SAP foram utilizadas.

```
…
<sap-extn:foreach xmlns:sap-extn="http://www.sap.com/xi/extensions"
name="Dynamic_Sequential_processing">
      <variables>
        <variable name="Response"/>
        <variable name="Receiver"/>
      </variables>
      <sequence>
        <invoke name="send_message"/>
        <receive name="Receive_response"/>
      </sequence>
    </sap-extn:foreach>
…
```

Figura 114 – Código XML de um processo BPEL em plataforma SAP

A.1.5.5 Atributos ARIS BPEL para execução em plataformas específicas

Outra forma de adicionar informações de fornecedores específicos ao padrão BPEL é a utilização de novos atributos a uma atividade BPEL existente. Mas, fique atento ao fato de que processos BPEL contendo tais atributos adicionais não são interoperáveis, ou seja, eles ficam dependentes da plataforma de execução alvo e, portanto, não podem ser reutilizados em plataformas de outros fornecedores.

O ARIS BPEL possui uma maneira de especificar esses atributos de execução específicos para uma determinada plataforma (ou EPSA – *Execution Platform-Specific Attributes*) e eles são considerados durante a exportação do ARIS BPEL/WSDL, sendo escritos no arquivo XML resultante.

O EPSA consiste em um conjunto de atributos definidos pelos usuários. Por padrão, o conjunto começa com o atributo 'User attibute text 150' no grupo de atributos 'Free attributes' e termina com 'User attribute text 169'. Quando a exportação BPEL inicia usando uma interface macro, o intervalo e tamanho dos sub-conjuntos a serem considerados podem ser definidos pelo usuário. Essa abordagem permite a especificação de uma variedade de plataformas diferentes, mesmo que apenas um processo BPEL esteja disponível. Desses atributos definidos pelo usuário, o formulário em pares 'nome-valor' EPSA será correspondido com uma representação no arquivo XML. O número de pares por objeto é determinado pelo intervalo de sub-conjuntos especificados. Se nada tiver sido definido, haverá 10 pares por objeto. A Figura 115 seguinte ilustra esse princípio: Se assumirmos que um atributo, por exemplo, 'User attibute text 150', contém o valor 'Stepname' e o 'User attribute text 151' contém o valor 'Name1', o par 'Stepname=Name1' será exportado.

Pelo fato de os atributos poderem ser renomeados no ARIS, filtros dedicados são oferecidos para a plataforma alvo. EPSA pode ser especificado pelos elementos BPEL.

- Activity, incluindo 'Extension'
- Variable
- Correlation

- PropertyAlias
- ProcessStart

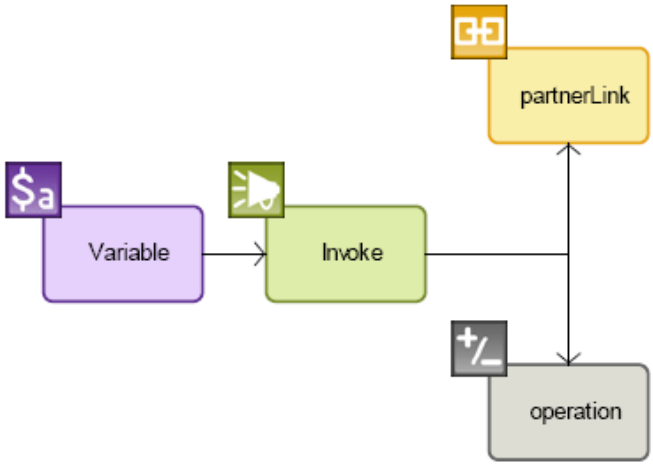


Figura 115 – Parte de um processo independente de plataforma