Eduardo Marreto Mendes

Características de Serviços Compostos para Desenvolvimento de Arquiteturas Orientadas a Serviços (SOA)

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para conclusão do curso de MBA em Tecnologia de Software.

São Paulo

2013

Eduardo Marreto Mendes

Características de Serviços Compostos para Desenvolvimento de Arquiteturas Orientadas a Serviços (SOA)

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Profa. Dra. Selma Shin Shimizu Melnikoff

São Paulo

2013

FICHA CATALOGRÁFICA

Mendes, Eduardo Marreto
Características de Serviços Compostos para desenvolvimento de Arquiteturas Orientadas a Serviços (SOA)/ E.M. Mendes. -- São Paulo, 2013.
59 p.

Monografia (MBA em Tecnologia da Informação) - Escola Politécnica da Universidade de São Paulo. Programa de Educa-ção Continuada em Engenharia.

1. Arquitetura Orientada a Serviços I. Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II. t.

DEDICATÓRIA

A Deus, familiares e amigos que tornaram mais este momento possível.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela saúde e motivação em conduzir este e todos os trabalhos em minha vida.

Agradeço aos meus colegas, docentes e amigos do programa de Pós-Graduação em Tecnologia de Software - PECE por terem me auxiliado nas disciplinas e me apoiado até a entrega deste trabalho.

A minha orientadora e amiga Selma, por sempre ter me apoiado pacientemente durante o desenvolvimento desta monografia, e soube compreender a minha rotina e ritmo de trabalho.

Aos meus pais e familiares que sempre me apoiaram em toda as fases e acontecimentos da minha vida.

Aos meus colegas de trabalho, por compartilharem materiais, experiências e compreensão nas horas que tive que dedicar a esta monografia e disciplinas obrigatórias.

Aos meus amigos que sempre me motivaram, dividiram angústias e compreenderam minha indisponibilidade para dedicar horas as atividades da pós.

RESUMO

Com base em referencial acadêmico e prática de mercado, este trabalho visa a apresentação das principais características de desenvolvimento de serviços compostos em Arquiteturas Orientadas a Serviços (SOA). São apresentados os principais conceitos relacionados à SOA, a importância do alinhamento de serviços com processos de negócio e características de desenvolvimento de composição. Do ponto de vista funcional, tem-se as características relacionadas com identificação e seleção de serviços visando reuso. Do ponto de vista técnico são apresentadas as principais características com enfoque em controle de invocações, tratamento de erros, transformação de dados, manutenção de estado e tarefas humanas em fluxos de composição de serviços. Também como enfoque, estas características são identificadas em uma ferramenta selecionada de mercado, verificando assim a aderência das características acadêmicas em um ferramental, e propiciando a desenvolvedores SOA uma visão prática das recomendações acadêmicas.

ABSTRACT

Based on academical references and Information Technology market practices, this work focus in the presentation of main caracteristics of composite services in Service Oriented Architectures (SOA). Concepts related with composite services are presented such as: SOA definition, service and composite service definitions, orquestration and SOA alignment with business process. After the conceptualization, main caracteristics of composite services are presented. In functional view, the approach of service identification and selection focusing service reuse. In technical view main caracteristics focusing service invoke control, error handling, data transformation, statefull service and human tasks in composite services flow are presented. Based on these approach, the caracteristics are identified in a selected market tool, and with this, denote the aderence of the academical service composition caracteristics in a market tool, and making it easy to SOA developers to attend academical recommendations.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SOA	*Service Oriented Architecture*
BPM	*Business Process Modeling*
BPMN	*Business Process Modeling Notation*
WSDL	*WebService Description Language*
BPEL	*Business Process Execution Language*
SLA	*Service Level Agreement*
XML	*eXtensible Markup Language*
XSL	*eXtensible Stylesheet Language*
XSLT	*eXtensible Stylesheet Language for Transformation*
JSP	*Java Server Pages*

SUMÁRIO

1. INTRODUÇÃO

Este capítulo apresenta as motivações acadêmicas e de mercado relacionadas à Arquitetura Orientada a Serviços, bem como o objetivo e as justificativas para o desenvolvimento deste trabalho.

A Arquitetura Orientada a Serviços (SOA – *Service Oriented Architecture*) tem sido amplamente discutida e fundamentada na literatura acadêmica da área de Tecnologia da Informação. Esta preocupação acadêmica está diretamente pautada no interesse de mercado sobre as técnicas de desenvolvimento voltado a serviços e sobre a evolução das ferramentas de mercado que acompanham as expectativas corporativas. O interesse de mercado está pautado em três vantagens competitivas de SOA: eficiência e alinhamento da Tecnologia da Informação aos objetivos estratégicos corporativos, aumento da adaptabilidade tecnógica a novas ferramentas de mercado e redefinição de processo de desenvolvimento com base em técnicas de reuso (ERL, 2010).

Todavia, apesar da evolução das ferramentas e processos, o desenvolvimento de serviços, que acompanhe as atividades funcionais de negócio (FAREGHZADEH, 2009) e que denote o dinamismo esperado pelas áreas de negócio (BIERBESTEIN, 2008), não se mostra uma atividade trivial.

A construção de serviços e alinhamento com modelagens de negócio monstram-se como atividades desafiadoras. A complexidade é aumentada se os serviços a serem desenvolvidos forem compostos, ou seja, se estes serviços controlam outros e coordenam um fluxo funcional.

Com base na complexidade dos serviços compostos em SOA, este trabalho apresenta as características de composição pautados da modelagem de negócio até a obtenção de características técnicas de desenvolvimento definidas por Thomas Erl (2010).

Para apresentação prática, as características técnicas de composição são identificadas e aplicadas em uma ferramenta de mercado para desenvolvimento.

Por fim, é feita a conclusão, levando como critério a aderência do ferramental de mercado as características estudadas e propostas pelas referências acadêmicas, e apresentando contribuições alcançadas e possíveis trabalhos futuros.

1.1. Motivação

A SOA possui atualmente ferramentas consolidadas, materiais bibliográficos e referências acadêmicas, buscando, com isso, maior eficiência no desenvolvimento de software através das práticas de reuso, do maior alinhamento com áreas de negócio e da maior possibilidade de integração com novas tecnologias de mercado.

Apesar de hoje existir maior maturidade ligada à tecnologia SOA, o desenvolvimento de serviços apresenta complexidade desde sua modelagem de negócio (FAREGHZADEH, 2009) até a obtenção de características técnicas (ERL, 2008). Se em níveis granulares já há complexidade no desenvolvimento, quando os serviços a serem desenvolvidos formam composições (serviços compostos) agregando diversos serviços, transformações e definição de um fluxo funcional, a complexidade aumenta mais.

Dado o desafio de modelagem, encontram-se, na literatura e no meio acadêmico, trabalhos voltados à identificação de características de serviços compostos. Nos trabalhos desenvolvidos por Erl (2010), as composições são obtidas através da aplicação de técnicas apropriadas, como manutenção de estado, interação entre serviços, tratamento de erros e análise de desenvolvimento com base em atividades de negócio. Porém não é fácil conciliar as características de composição em ferramentas de mercado e ter a visualização clara de sua aplicação.

A abordagem de técnicas de desenvolvimento em ferramentas de mercado, aliadas às características de referência acadêmica visa permitir uma maior maturidade na elaboração de serviços compostos.

1.2. Objetivo

Este trabalho tem por objetivo a identificação das principais características de dos serviços compostos, desde a identificação e seleção de serviços até o desenvolvimento. Tem-se também, como objetivo, a identificação e a análise das características de composição de serviços, do ponto de vista acadêmico, em uma ferramenta de mercado, propiciando assim uma análise de aderência destas características e possibilitando uma visão prática de suas aplicação aos desenvolvedores SOA.

1.3. Justificativa

Na literatura, as principais propostas existentes para características de serviços compostos são as de ERL (2010), sendo complementado, neste trabalho, com os trabalhos de Fareghzadeh (2009), Bierbestein (2008), Papazoglu e Van Den Heuvel (2006), Silva (2010) e com materiais de fornecedores de Software como Oracle (2012) e IBM (2012).

Erl (2010) define os serviços compostos como um fluxo lógico de serviços compostos por outros serviços, apontando que seu desenvolvimento envolve características técnicas e funcionais.

As características funcionais envolvem o alinhamento com processos de negócio, citadas também por Fareghzadeh (2010) como fundamentais para a identificação e seleção de serviços. Silva (2010) menciona que para se ter visão sobre fluxos de negócio, o desenvolvimento de composição deve ter como insumo diagramas de modelagem de negócio (BPMN), remetendo assim a técnica de identificação de serviços baseado em abordagem top-down (YANAI, 2010).

Para as características técnicas, Erl (2010) aponta as características de invocações a serviços com diferentes contextos de negócio, tratamentos de erros e controles do fluxo para garantir a integração e desenvolvimento do workflow de composição.

Materiais de fornecedores como Oracle (2012), IBM (2012) e Red Hat representado por DiMaggio (2012), apontam as práticas de desenvolvimento de serviços seguindo as melhores práticas de desenvolvimento, motivando a apresentação das características técnicas de Erl (2010) para desenvolvimento de composição aplicando as técnicas em ferramental de mercado.

1.4. Estrutura do Trabalho

O Capítulo 1, Introdução, apresenta a motivação, o objetivo, a justificativa e a estrutura do trabalho.

O Capítulo 2, Arquitetura Orientada a Serviços, apresenta os principais conceitos de SOA e serviços, e o referêncial bibliográfico associado ao trabalho.

O Capítulo 3, Características de Serviços Compostos, apresenta as atividades de Identificação e Seleção de Serviços, bem como suas características técnicas de desenvolvimento de composição.

O Capítulo 4, Aplicação das Características de Serviços Compostos, apresenta as características de serviços compostos (definidas no capítulo 3) em uma ferramenta de mercado com boa avaliação por Gartner (2012).

O Capítulo 5, Considerações Finais, apresenta a conclusão do trabalho, bem como a contribuições alcançadas e possíveis trabalhos futuros.

2. ARQUITETURA ORIENTADA A SERVIÇOS

O objetivo deste capítulo é apresentar os conceitos sobre SOA e contextualizar conceitos, terminologias e conteúdos teóricos que permeiam o tema composição de serviços.

2.1. Conceito sobre Arquitetura Orientada a Serviços (SOA)

A Arquitetura Orientada a Serviços (SOA) é um estilo de arquitetura de software com objetivo de fornecer funcionalidades de aplicação em forma de serviços e também uma maior pluralidade tecnológica (por meio de conectores e interação com serviços parceiros externos) (ERL, 2010).

Ainda segundo este autor:

["SOA é habilitador arquitetural para rápida resposta a mudanças de negócio e o efetivo reuso de ativos faz com que SOA seja um modelo que clame por eficiência, agilidade e produtividade da corporação. Por posicionar serviços de forma primária, SOA remete ao suporte às realizações de objetivos estratégicos corporativos de negócio associados às áreas de Tecnologia da Informação..."]

(ERL, 2010)

["Como forma de arquitetura tecnológica, uma implementação SOA pode consistir de uma combinação de tecnologias, produtos, aplicações, variações de infraestrutura e várias outras partes..."]

(ERL, 2010)

A configuração técnica de SOA se dá através de um barramento de serviços centralizado (*Enterprise Service Bus*) com capacidade de desenvolvimento e adaptação a múltiplas origens e destinos de dados (objetos de banco de dados, *web services*, filas e arquivos de dados). Acoplado ao barramento de serviços tem-se o ferramental de interface, operação e execução de fluxos de composição de serviços (BPEL - Workflows) (Figura 1).

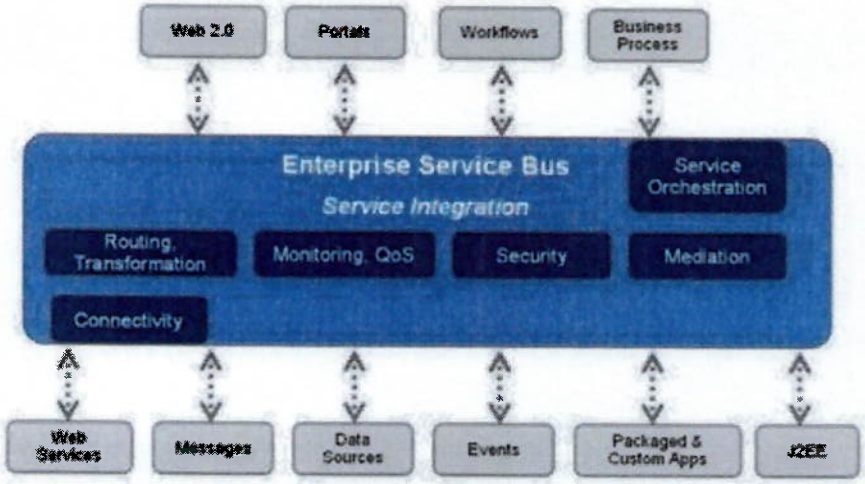

Figura 1. Arquitetura Técnica para Infraestrutura de aplicação focada em SOA (Erl, 2010)

Com o uso de um barramento de serviços, tem-se a centralização dos serviços e capacidade de integração de múltiplas origens de dados. Quanto ao desenvolvimento de serviços, a centralização permite a agregação e orquestração funcional de serviços, provendo, então, a capacidade de desenvolvimento de serviços compostos e *workflow* (ORACLE[2], 2012).

2.2. Serviço

2.2.3 Definição de serviços

Sistemas computacionais possuem enfoque na automatização e possibilitam a execução de atividades em distintas áreas. No passado, a execução destas atividades se dava de forma independente, não havendo a interação entre áreas, sistemas e parceiros de negócio.

Todavia, com a evolução computacional e, consequentemente, com maior exigência e expectativa nas áreas de Tecnologia da Informação, foi necessário que os sistemas se integrassem, surgindo o conceito de serviço de software.

Um serviço pode ser colocado como uma unidade de uma solução lógica envolvendo sistemas (ERL, 2008). Cada serviço representa a interação entre um sistema provedor e um ou vários sistemas consumidores (Figura 2).

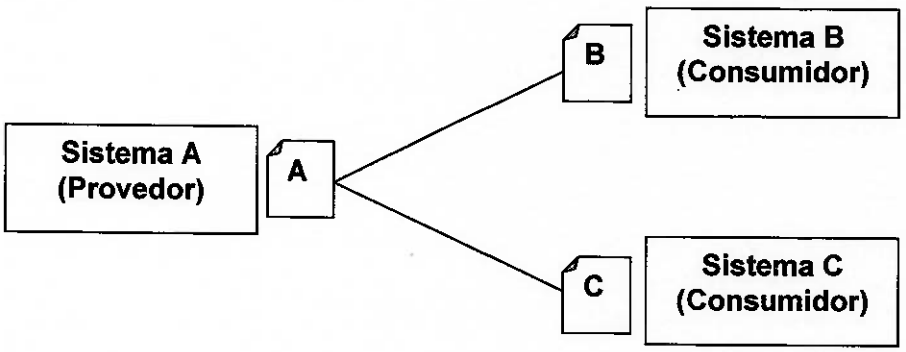

Figura 2. Serviços entre sistemas e contratos

Dados seus objetivos primários, serviços podem ser caracterizados dentro de contextos funcionais, como partes de atividades funcionais, por exemplo, cadastro de um determinado cliente, ou técnicos, como células de uma atividade técnica, por exemplo, envio de log.

Porém, independente do contexto técnico ou funcional, um serviço exige uma regra que regularize a comunicação entre os sistemas (FAREGHZADEH, 2010) denominada de contrato de serviços. Os contratos visam descrever o conjunto de dados, métodos e técnicas implementados para a comunicação entre os sistemas.

Do lado do sistema provedor, o contrato denota todas as capacidades que o sistema origem fornece para o serviço. De forma complementar, o contrato do sistema consumidor denota todas as capacidades que o sistema destino espera do serviço.

Apesar de conceitualmente lógico, os serviços de software culminam em uma série de atividades complexas quanto à governança e implementação técnica. Uma das atividades mais importantes e complexas no ciclo de vida de serviços é a fase de Identificação de serviços.

2.2.3 *Análise e Identificação de serviços*

A identificação de serviços consiste na atividade de análise da atividade de negócio a ser automatizada e, a partir do modelo de negócio, definir os serviços candidatos dentro do contexto de negócio definido.

Esta atividade é uma das principais e mais complexas em uma solução pautada em serviços e, como ponto de atenção, erros efetuados durante a identificação podem ser propagados a outros serviços que farão uso deste serviço em desenvolvimento (FAREGHZADEH, 2009).

Segundo (PAPAZOGLU; VAN DEN HEUVEL, 2006), a meta principal da atividade de identificação é a obtenção de serviços corretos visando automação de processos de negócio.

Durante a fase de identificação, as principais atividades são:

- Identificação de processo de negócio: visa a identificação dos serviços que precisam ser agregados em um processo de negócio, a partir da abstração das atividades de negócio;
- Definição de escopo: define as atividades e interações que estão no escopo do processo de negócio;
- Análise de *gap* de negócio: avalia os serviços existentes no portifólio da organização para fazer o mapeamento com as necessidades dos processos de negócio;
- Realização do serviço: avalia cenários de realização para verificar custos, riscos, benefícios etc.

Há diferentes estratégias para identificação de serviços candidatos e, para cada um deles, há benefícios e desvantagens. Como enfoque deste trabalho, são apontadas as abordagens *botton-up* e *top-down*.

2.2.3 *Processo de Identificação Top-down*

Diante da atividade de Realização do Serviço, a abordagem *top-down* (ou decomposição de domínio), consiste na análise das áreas funcionais e o subsistemas que fazem parte de um domínio e, depois, da análise dos processos de negócio e dos casos de uso de negócio (YANAI, 2010).

O objetivo desta abordagem é a obtenção de serviços candidatos a partir de modelos de negócios, partindo de diagramas funcionais a componentes funcionais e técnicos.

Partindo de análise, tem-se as notações de fluxo de negócio (representados, por exemplo, através de BPMN – *Business Process Modeling Notation*), requisitos, especificações funcionais, casos de uso, especificações de serviços (componentes).

2.2.3 Processo de Identificação Botton-up

Como opção da abordagem *top-down*, a *bottom-up* consiste na análise da solução a patir dos componentes e recursos existentes com potencial de reuso. Estes componentes, caso já não sejam serviços, são considerados como serviços candidatos (YANAI, 2010).

O objetivo desta abordagem é atingir a solução de tarefas dentro do modelo de negócio pretendido a partir de componentes.

A análise para o desenvolvimento de serviços é iniciada a partir dos portifólios de serviços, casos de uso, requisitos e diagramas de fluxo de negócio (representados, por exemplo, através de BPMN – *Business Process Modeling Notation*).

2.3. Serviços Compostos

A definição de serviços compostos se dá de forma intuitiva, como serviços compostos de serviços. Segundo (ERL, 2008) se algo é decomposto, este pode ser recomposto e o objetivo desta quebra se dá porque há mais benefícios diretos no uso de células menores, ao invés do uso apenas uma grande célula.

Do ponto de vista computacional, a decomposição de soluções proporcionou maior escalabilidade das soluções lógicas implementadas. Esta tendência influenciou diretamente a criação de paradigmas de programação, como é o caso da Orientação a Objetos, em que a lógica de aplicação é separada em classes e métodos (ERL, 2008).

No contexto de composição, os serviços assumem diferentes papéis, podendo ser o serviço controlador ou o serviço membro da composição.

2.2.3 Serviço controlador

O serviço controlador é a cabeça na hierarquia de composição (ERL, 2008). Esta definição se dá pois o serviço controlador contém toda a lógica de *workflow*, tratamento de exceções e chamadas dos demais serviços.

Um serviço controlador pode ser composto de vários serviços controladores (sendo estes chamados de serviços sub-controladores), todavia a composição de composição aumenta a complexidade de gestão e rastreabilidade de impactos a mudanças de serviços que compõem a solução (ERL, 2008).

A figura 3 ilustra o conceito de serviços controladores:

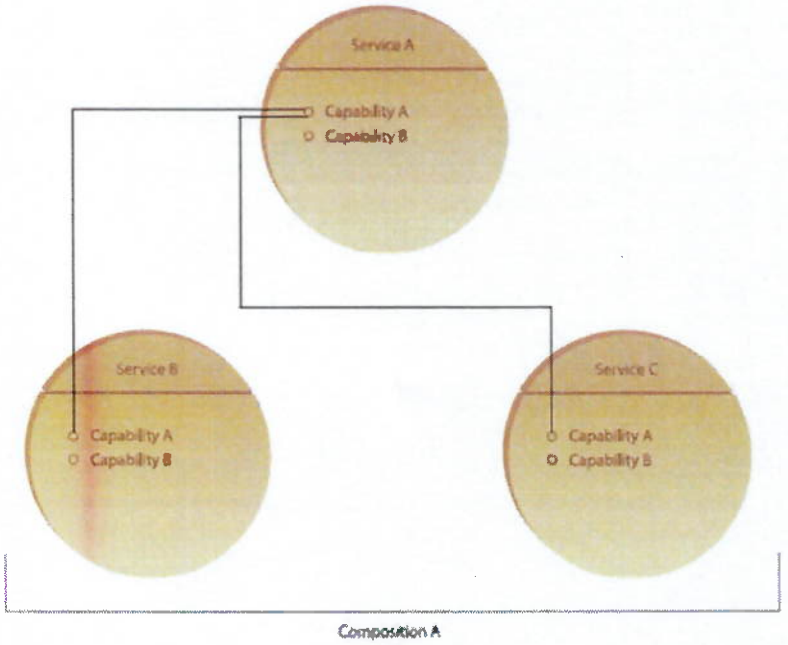

Figura 3. Ilustração de Serviços Controladores *(ERL, 2010)*

De acordo com a ilustração de (ERL, 2008), o serviço A é composto de dois serviços (B e C) cada um com duas capacidades funcionais. Para a capacidade funcional A, o serviço controlador faz uso das capacidades funcionais A dos serviços B e C. A mesma lógica pode ser aplicada para as capacidades funcionais B.

2.2.3 Serviço membro de composição

Um serviço membro de composição representa um serviço que faz parte de um serviço controlador.

Segundo (ERL, 2008), como notação de negócio, é mais comum que entidades de negócio (exemplo: cliente, fornecedor e venda) sejam identificadas como membros de composição e controladas pelos serviços identificados como tarefas funcionais de negócio (exemplo: comprar, vender e entregar).

A figura 4 ilustra os Serviços Membros de Composição:

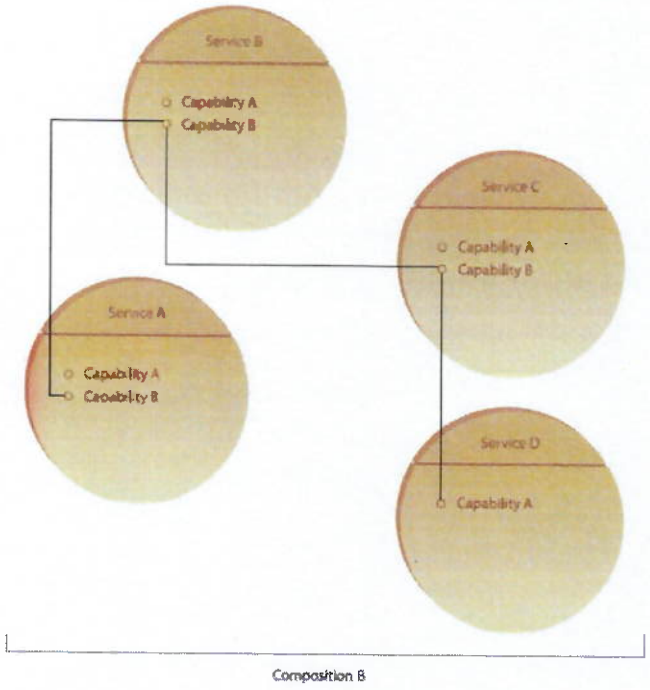

Figura 4. *(ERL, 2010)* Ilustração de Serviços Membro de Composição

Neste exemplo a capacidade funcional B do Serviço B é composto pelas capacidades funcionais B dos serviços A e C. De forma complementar a capacidade funcional B, do Serviço C (neste caso é um sub-controlador), é composto pela capacidade funcional A do Serviço D.

2.2.3 Tipos de serviços compostos

De acordo com (ERL, 2008) é dado o nome de serviços primitivos aos serviços compostos de poucos serviços, tipicamente quando a organização possui portifólio simplificado de serviços.

Exemplos de serviços primitivos são aqueles que um serviço controlador possui serviços componentes que aplicam regras ou operações simplificadas de dados e fazem retorno a um sistema origem ou serviço destino.

Aos serviços mais sofisticados com requisitos de automação de mais entidades funcionais de negócio é dada a definição de composições complexas de serviços (ERL, 2008).

Estes serviços exigem maior análise quanto ao *workflow* aplicado, entidades envolvidas e tarefas automatizadas.

2.4 Orquestração

A orquestração de serviços é uma técnica de desenvolvimento de serviços compostos, em que um processo de negócio é abstraído em forma de um serviço complexo (SILVA, 2010).

Seu conceito é baseado em quatro pilares conforme apresentado na figura 5 (ERL, 2010).

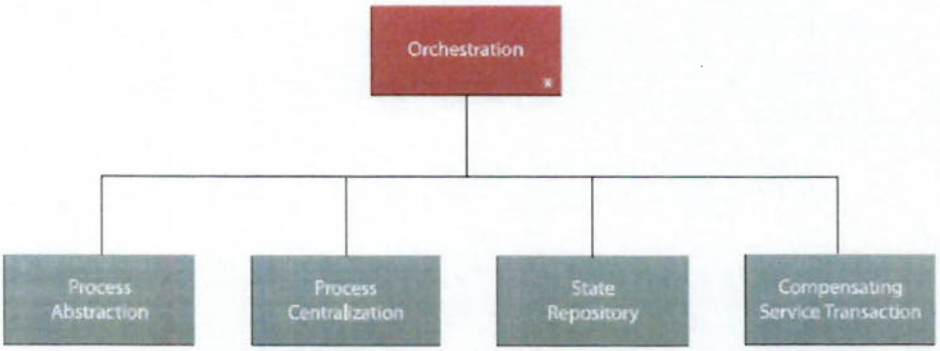

Figura 5. Pilares de Orquestração (ERL, 2008)

Tem-se, a seguir, a descrição doe elementos dessa figura:

- Abstração de Processo (*Process abstraction*): responsável pela aplicação das regras de processo somente no serviço ou componente de orquestração;
- Centralização de Processo (*Process centralization*): limita a distribuição física do processo em um único componente (o serviço orquestrado);
- Repositório de Estado (*State repository*): permite que os estados transacionais dos serviços sejam armazenados;

- Compensação de Transação de Serviços (*Compensation Service Transaction*): permite que, em caso de falha durante o fluxo de orquestração, haja compensação funcional da transação.

A idéia central da orquestração é a construção de um mecanismo que permite a comunicação de dois ou mais sistemas, utilizando um elemento denominado orquestrador central. Sua grande vantagem, é que não existe a necessidade de se refazer funcionalidade de sistema já existente, deixando a cargo do orquestrador central a recepção e o retorno das mensagens (SILVA, 2010). A figura 6 exemplifica uma estrutura orquestrada utilizando de forma específica integrações Web Services.

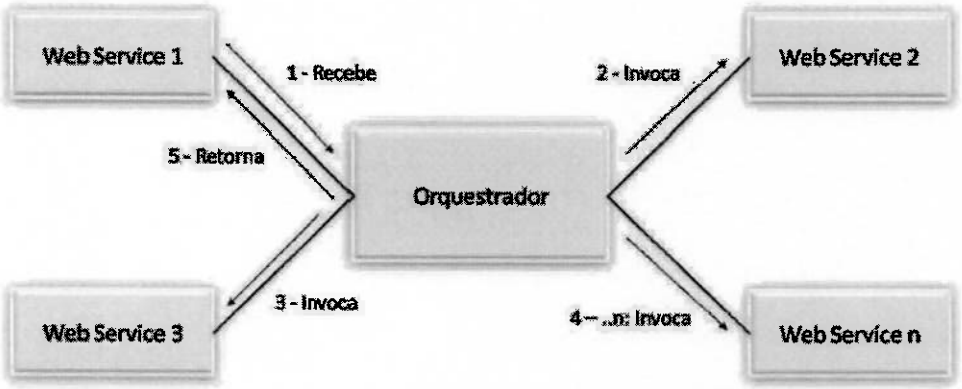

Figura 6. Esturura Orquestrada (SILVA, 2010)

Como alternativa a este mecanismo, tem-se a coreografia. Ao contrário da orquestração, os serviços conhecem uns aos outros e cada um conhece sua função dentro do fluxo do processo.

A coreografia é realizada através da troca de mensagens entre os serviços, como ilustrada na figura 7.

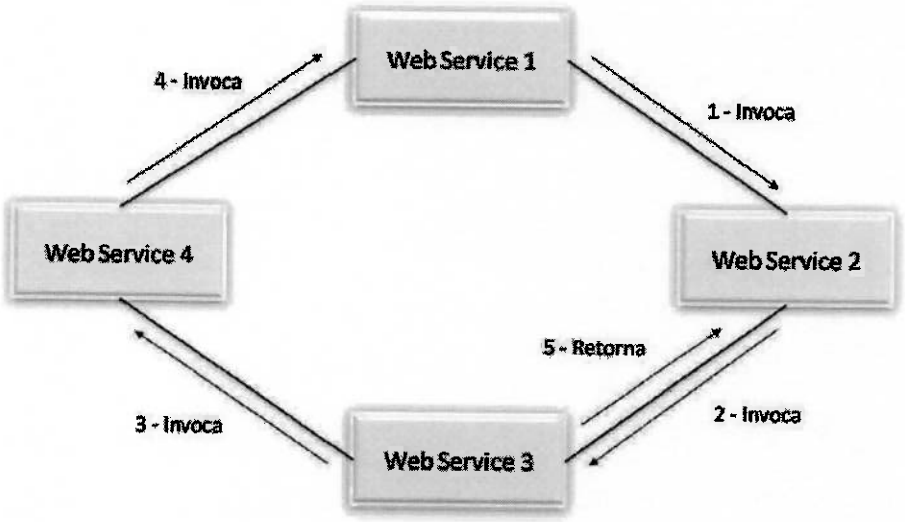

Figura 7. (SILVA, 2010) Conceito de Coreografia

Note-se que não existe, neste caso, um elemento central que coordene os demais serviços e cada um deles se comunica diretamente com os outros, seguindo uma coreografia lógica.

A prática de orquestração está fortemente associada a tecnologias de desenvolvimento com linguagem BPEL (SILVA, 2010). O desenvolvimento de fluxos e regras da orquestração está vinculado de forma direta à modelagem de processos e definições funcionais do processo automatizado.

2.5 SOA e Processos de negócio

A SOA deve estar diretamente associada à automatização e integração de processos de negócio, e deve-se ter a visão da funcionalidade que é implementada (Fareghzadeh, 2010).

Para isto, para a modelagem e implementação de serviços compostos, faz-se necessário ter a visão de modelagem de processos de negócio (BPMN) para a implementação baseada em ferramenta (BPEL) (SILVA, 2010).

2.2.3 *Workflow*

Seguindo a definição literal, um *workflow* (fluxo de trabalho) consiste em um caminho que um determinado trabalho é organizado, sendo constituído por estágios em um processo particular.

De forma mais específica, entende-se por *workflow* a formalização total ou parcial de um processo de negócio, representado através de participantes que realizam atividades, durante as quais os documentos são passadas de um participante para outro, de acordo com um conjunto de regras e procedimentos (MOLZ; THOM, 2010).

Um *workflow* é definido como uma coleção de tarefas organizadas para realizar um processo de negócio. Uma tarefa pode ser executada por um ou mais sistemas de computador, por um ou mais agentes humanos, ou então por uma combinação destes. O *workflow* define a ordem de execução das tarefas e as condições em que cada tarefa é iniciada.

Dentro do contexto SOA, um *workflow* se caracteriza como uma sequência realizada por um serviço controlador, que possui serviços membro de composição.

Os serviços membros de composição podem ser composições simples ou complexas (controladores invocando outros serviços controladores) ou podem apresentar interações com agentes humanos. Neste caso, tarefas humanas são componentes que permitem atividades tais como preenchimento de formulário, aprovações manuais ou complemento de informações que trafegam pelos serviços compostos, através de telas.

Os principais elementos de *workflow* segundo (MOLZ; THOM, 2010) são os seguintes:

- Atividade: é a descrição de uma parte de trabalho que forma um passo lógico dentro do processo. Ex.: aprovação de uma ordem de compra.
- Instância de Atividade: é a representação da ocorrência de uma atividade em um processo.
- Participante do *workflow*: é aquele que executa uma atividade.
- Lista de trabalho (*worklist*): é uma lista de atividades associada a um determinado participante de *workflow* ou a um grupo de participantes que podem compartilhar uma lista de atividades comum.

- Papel: é um conjunto de participantes que possuem um mesmo leque de características (habilidades) que os tornam aptos a executarem a atividade relacionada ao papel.
- Gatilho: um evento E dispara uma atividade A, se a ocorrência de E causa a execução de A. Deve-se observar que E pode ser um evento, uma atividade ou um participante, mas A é sempre uma atividade.
- *And-Split*: representa a ocorrência de um conjunto de atividades B, C e D que são ativados paralelamente, apenas após a execução de uma atividade A;
- *And-Join*: representa a ocorrência de uma dada atividade D que é ativada apenas após a execução de todas as atividades A, B e C.
- *Or-Split*: representa a ocorrência de um conjunto de atividades B, C ou D que ocorre após o término de uma dada atividade A.
- *Or-Join*: representa uma atividade D que ocorre após a execução da atividade A, B ou C.

Há diversas modelagens para a representação de *workflows*. Uma das mais utilizadas, tanto na academis quanto no mercado, é a BPMN (*Business Process Modeling Notation*), que é utilizada neste trabalho.

2.2.3 Business Process Modeling

BPM (*Business Process Management* – Gestão de Processos de Negócio) visa a análise, a definição, a execução, o monitoramento e a administração de processos para definir, então, a automação de processos corporativos (OMG, 2012).

Um dos objetivos básicos é a obtenção de padronização de processos corporativos para melhorar produtividade e eficiência.

Segundo OMG (2012) a gestão de processos, em diferentes companhias atingem diferentes níveis de maturidade. Sua avaliação é feita através de BPMM (*Business Process Modeling Maturity*), cujos níveis são:

- Inicial: quando os processos são inconsistentes e os resultados não são previsíveis;
- Gerenciado: quando os processos são capazes de serem reutilizados e potencializa o comprometimento de grupos de trabalho; no entanto, os grupos

de trabalho utilizam diferentes procedimentos para realizar atividades similares;

- Padronizado: quando as melhores práticas são identificadas nos processos e estas são replicadas para diferentes processos para atender diferentes necessidades de negócio. Neste nível há uma maior eficiência devido à replicação de procedimentos e ao aprendizado das práticas de métricas e experiência;
- Previsível: quando os processos, além de estarem padronizado possuem métricas e controle de variação, podendo ser previsíveis em estados intermediários;
- Inovativo: quando práticas pró-ativas e ações oportunas são tomadas em conjunto para solucionar *gaps* corporativos que impeçam o *modus operandis* a atingir os objetivos estratégicos.

O BPM utiliza a modelagem de processos de negócio como ferramenta para definição e avaliação de processos. Para isso, os processos são modelados através de BPMN (*Business Process Modeling Notation*) e, a partir dos modelos, pode-se realizar a automação de processos, definindo-se os serviços compostos controladores.

2.2.3 BPMN

A BPMN é uma notação gráfica utilizada para representar papéis e fluxos de atividades e objetos dos processos de negócio (OMG, 2012).

Os papéis são designações de partes do processo para usuários ou áreas específicas e são representados através de *swimlanes,* sendo que as atividades são representados na forma de fluxograma. Um exemplo de diagrama BPMN está representado na figura 8.

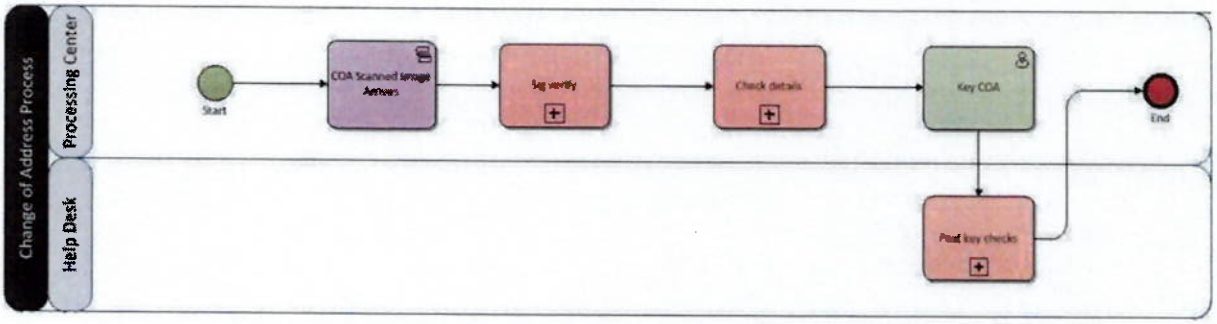

Figura 8. Exemplo OMG de Diagram BPMN (OMG,2012)

O exemplo da figura 8 apresenta o fluxo de recebimento de documentos entre dois setores: *Processing Center* e *Help Desk*.

De forma simplificada, um diagrama BPMN é constituído por seguintes componentes (OMG, 2012):

- Componentes de fluxo: são componentes centrais que representam o comportamento do processo de negócio. Os componentes de fluxo são representados por eventos, atividades e *gateways*;
- Componentes de dados: são representados por dados gerados no fluxo, dados de entrada, dados de saída e dados armazenados;
- Componentes conectores: são responsáveis pela ligação entre componentes de fluxo; os principais tipos os conectores são conector de sequência, mensagem, associação e associação de dados;
- *Swimlanes*: são os separadores lógicos dos responsáveis pelas atividades, podendo ser representados por *pools* ou linhas;
- Artefatos: são utilizados para fornecer informações adicionais aos processos. Atualmente a notação oficial aponta o uso de artefatos de grupos e anotações de textos.

Maiores detalhes sobre BPMN podem ser encontrados em (OMG, 2012).

2.2.3 BPEL

BPEL (*Business Process Execution Language*) é uma linguagem técnica, baseada em BPMN, utilizada para definição e execução de um processo de negócio através da orquestração, mais comumente utilizada com integrações *web services*. Permite, de forma simplificada, a realização de SOA através de uma abordagem *top-*

down para composição, orquestração e coordenação de *web services*. (MOREIRA et al., 2011).

Um processo BPEL é exposto como um serviço (*web service*) que compõe serviços existentes e fornece acesso às operações destes serviços. Para invocar um processo BPEL deve-se invocar o serviço resultante da composição.

Como principais componentes para descrição de fluxos BPEL tem-se (MOREIRA et al., 2011):

- *Partner Links*: são chamadas a *web services* parceiros;
- Variáveis: são variáveis aplicadas aos dados que trafegam pelo fluxo, possibilitando o uso de funções de transformação;
- *Invoke*: invoca uma operação de um serviço;
- *Receive*: recebe a mensagem de uma fonte externa;
- *Reply*: envia uma resposta para uma fonte externa;
- *Waiting*: realiza uma pausa por um período especificado;
- *Assign*: copia dado para variável;
- *Throw*: deteta erros na execução do processo;
- *Terminate*: finaliza a execução de uma instância do serviço;
- *Compensate*: desfaz alterações em caso de erro.
- *Sequence*: define a ordem de execução;
- *Switch*: é um recurso para lógica condicional;
- *While*: é um recurso para laços (*loops*);
- *Pick*: possibilita o bloqueio do fluxo no processo até que um determinado evento ocorra (usualmente a recepção de uma mensagem) ou ocorra fim de temporização (*timeout*);
- *Flow*: é um recurso para fluxos paralelos;
- *Scope*: define um escopo para agrupamento de atividades, criação de variáveis locais, tratamento pelo mesmo manipulador de erro (*fault-handler*) e deteção de exceções.

2.6 Considerações do capítulo

Com base nas definições apresentadas e nos referênciais acadêmicos, este trabalho toma, como base, a apresentação de características de Serviços Compostos com técnicas de Orquestração, usando como guia a orientação de modelagem *top-down*. O desenvolvimento é feito a partir de modelos de negócio representados em BPMN até a definição técnica dos serviços.

Com base em modelagens de processo de negócio, teoria de conceituação SOA e conhecimento de linguagem BPEL, faz-se possível o desenvolvimento de serviços compostos.

3. CARACTERÍSTICAS DE SERVIÇOS COMPOSTOS

Depois de realizada a conceituação teórica, este capítulo inicia o trabalho de apresentação das principais características para desenvolvimento de serviços compostos.

Baseada na modelagem BPMN e na abordagem *top down*, as características são descritas através das atividades de identificação de serviços, seleção dos serviços membro da composição e, com maior enfoque, em composição de serviços, considerando as características de implementação do fluxo interno do serviço.

3.1. Definição de serviços compostos aderentes aos processos de negócio

Considerando a automação de processos de negócios através de serviços, o fluxo de composição tem, como principal artefato de entrada, os modelos BPMN das tarefas funcionais.

Com base no fluxo de composição, obtido a partir dos modelos de processo de negócio e no conhecimento dos serviços que devem fazer parte da composição, é realizada a identificação e seleção dos serviços existentes em repositório de serviços. Se for necessário, são desenvolvidos novos serviços, a fim de atender às necessidade do fluxo composto. Após a seleção e desenvolvimento dos serviços membros de composição, projeta-se o fluxo composto, considerando as características de interação entre os serviços membro e possíveis tratamentos de dados e execução no escopo do fluxo composto.

3.2.3 Identificação de serviços candidatos baseado em modelos BPMN

Para a definição de serviços, com abordagem top-down, os modelos de processos de negócio (diagramas BPMN) são os principais insumos. A atividade primária é a identificação dos serviços funcionais que farão parte da automação do processo selecionado, a partir desses modelos.

Quando se automatiza um processo de negócio, os principais serviços candidatos são os componentes de fluxo do diagrama BPMN, constituído por

eventos, atividades e *gateways*. Além desses, é necessário um orquestrador que controle os serviços componentes.

A figura 9 representa um fluxo BPMN composto por raias de consumidor, vendedor, montador, suporte e fornecedor.

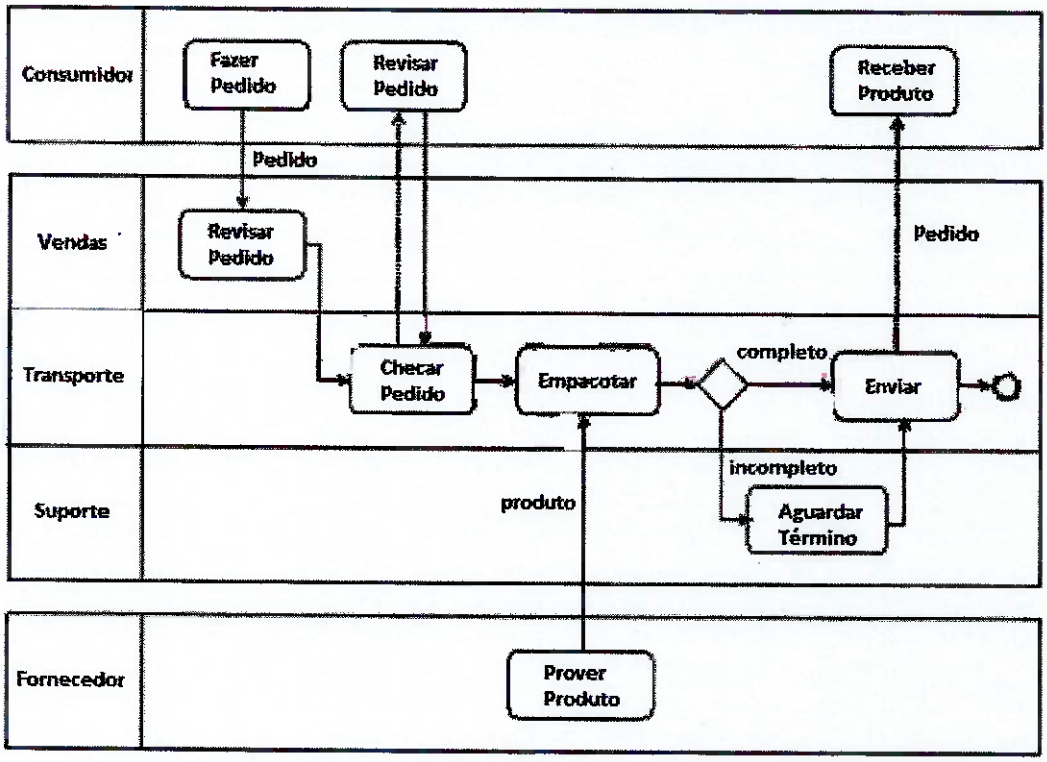

Figura 9. Exemplo de processo modelado BPMN

Todas as atividades do diagrama são caracterizadas como operações de serviços candidatos e devem ser avaliadas durante a fase de seleção de serviços membros da composição. Os fluxos entre as atividades também são considerados para definir a lógica de composição de serviços.

Além disso, é importante também ter a visão dos dados que trafegam através dos elementos do diagrama. Segundo Faregzadeh (2009), todo serviço deve gerar um contrato que inclui os dados e, para as fases seguintes de seleção de serviços, é importante o conhecimento dos contratos necessários para o fluxo.

3.2.3 Seleção de serviços membros da composição

A partir da seleção dos serviços candidatos selecionados, é feita a análise de reuso dos serviços com base em repositórios corporativos e centralizados de serviços. Os serviços são descritos nos respositórios através de formulário com os dados dos seguintes tipos (KRAFZIG; BANKE; SLAMA, 2004):

- Contexto funcional;
- Nome do serviço, operações e argumentos;
- Dono do serviço (contexto no negócio e desenvolvedor);
- Contrato do Serviço (tanto para requisição quanto para a resposta);
- SLAs (*Service level agreement* – Acordo de nível de serviço);
- Definições de segurança e acesso.

A decisão de reuso de serviços se faz através da análise de características exigidas pelo fluxo composto e as características dos serviços catalogados no repositório. Se o serviço não for encontrado no repositório ou se os existentes não atendem às características funcionais e/ou técnicas, um novo serviço deve ser desenvolvido. A análise e decisão de reuso de serviços deve ser da responsabilidade de uma equipe de arquitetura corporativa e governança de serviços.

Após a seleção e o desenvolvimento de todos os serviços membros de composição, faz-se o desenvolvimento do fluxo do serviço orquestrador.

3.2. Desenvolvimento do fluxo de Atividade do serviço orquestrador

Além da funcionalidade apontada no diagrama de processo de negócio, os desenvolvedores de composição devem atentar-se às características técnicas de serviços. As características de invocação e repostas de serviços, manutenção de estado transacional, interação humana, transformação de estruturas de dados, roteamento e tratamento de erros são fundamentais para o desenvolvimento e atendimento de requisitos do fluxo funcional (ERL, 2010).

3.2.3 Invocações e respostas de serviços

As invocações e respostas de serviços compreendem as propriedades de sincronismo entre o serviço controlador, neste cenário como invocador, e o serviço membro de composição.

As definições de sincronismo estão compreendidas entre o contrato de serviços invocados e a configuração de invocação de serviço. Como principais características de invocação tem-se: síncronas, assíncronas e *fire-and-forget*.

3.2.1.1 Invocações síncronas

Uma invocação síncrona é aquela em que o invocador requer uma resposta para dar continuidade ao fluxo operacional (DIMAGGIO, 2012).

Seu uso está diretamente ligado às requisições de *web services*, tendo por operações características as consultas de dados, em que o retorno da invocação é utilizado para decisões lógicas ou enriquecimento de mensagens.

Apesar de ser um método intuitivo de invocação de serviços, seu uso pode implicar em problemas de desempenho de fluxo, pois a invocação fica estacionada aguardando resposta e consumindo recursos, já que uma instância lógica do serviço composto tende a ficar mais tempo em execução.

3.2.1.2 Invocações assíncronas

Uma invocação assíncrona é aquela em que o serviço invocador exige uma resposta, porém há continuidade do fluxo enquanto esta não é entregue (DIMAGGIO, 2012).

A invocação assíncrona permite estruturas mais robustas e com melhor desempenho do que a síncrona, já que nenhuma resposta tende a bloquear o fluxo, permitindo assim o paralelismo.

Seu uso tem por características os *web services* com operações de resposta, adaptadores a filas de dados, em que haja resposta, e outros adaptadores a origem de dados que possuam resposta lógica quanto ao consumo de dados.

Apesar dos serviços assíncronos onstituirem uma boa prática quanto ao desempenho, a possibildade de dar continuidade ao fluxo, mesmo sem a resposta,

pode aumentar a complexidade no tratamento de erros e análise de dependência entre serviços.

Entende-se, por dependências funcionais em cenários de fluxo, as simulações de sincronização. Para estas simulações utiliza-se o método *pick,* presente em fluxos desenvolvidos em serviços compostos BPEL.

O método pick consite na adição de uma condicional de retorno para prosseguimento do fluxo. Em uma visão de execução, é como se houvesse um fluxo síncrono, porém com chamadas assíncronas e uma estrutura de controle de fluxo, que força a espera de resposta para continuidade do fluxo (Figura 9). A condicional de retorno está presa no fluxo lógico pelo *callback* de retorno ou pelo *timeout* de resposta do serviço invocado.

A Figura 10 apresenta um fluxo com uso do método pick. Há uma subfluxo que necessita de um retorno para continuidade. Caso a mensagem não chegue, para que o fluxo não fique bloqueado, é colocado um contador que cronometra o tempo de aguardo de resposta.

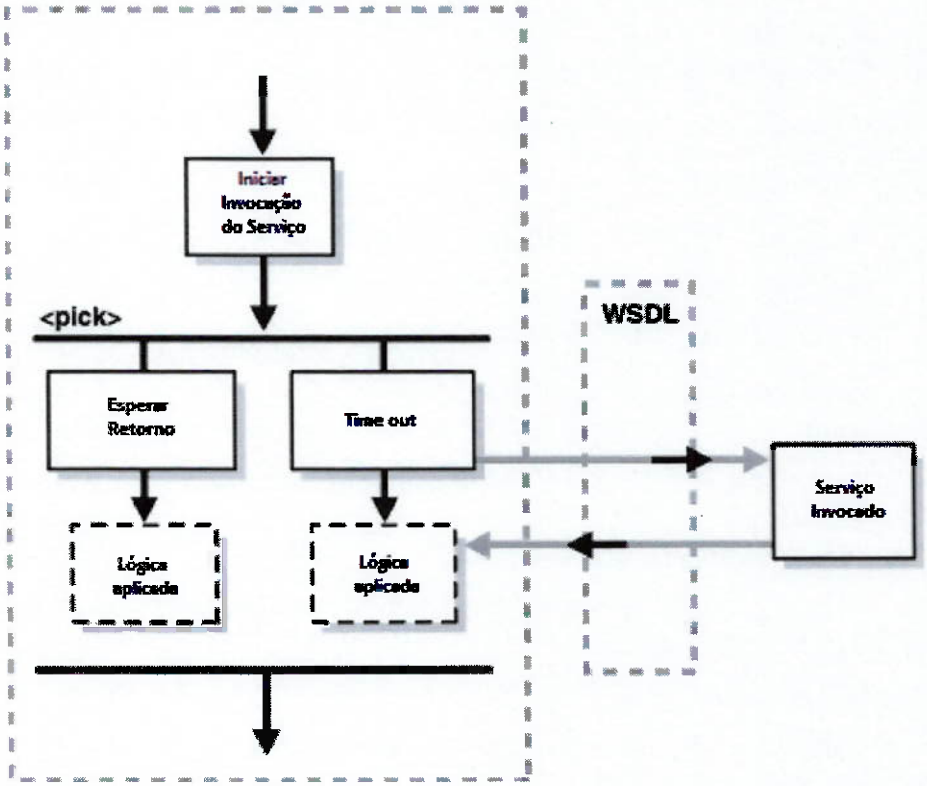

Figura 10. Funcionamento do componente *pick*

Nota-se que, na utilização do método *pick*, podem ocorrer as mesmas características de bloqueio de recurso de fluxo presentes na invocação síncrona, uma vez que se tem uma instância lógica parada em execução.

Como solução ao bloqueio de recurso, algumas ferramentas disponíveis implementam a desidratação da instância no servidor de aplicação, que consiste no armazenamento em memória do servidor e manutenção da instância lógica em aguardo, liberando recursos presos à execução (ORACLE, 2012). Assim que o serviço invocado retorna com a resposta, automaticamente o controle de desidratação re-hidrata a instância e permite a continuidade do fluxo, evitando-se assim a alocação de recursos de execução para o fluxo em aguardo.

3.2.1.3 Invocações Fire-and-forget

As invocações do tipo *Fire-and-forget* (também conhecidas como *one-way*) são aquelas que não espera nenhuma resposta do serviço invocado (DIMAGGIO, 2012).

Seu uso não implica em preocupações quanto ao tratamento de erros no fluxo, pois não há resposta, ou queda de desempenho. Tipicamente serviços do tipo *Fire-and-forget* possuem função de notificação (por exemplo, serviço de envio de notificação a *e-mail*, notificação via SMS e outros) ou serviços com garantia de persistência da mensagem, como o uso de filas de dados.

3.2.2 Transação com manutenção de estado (Statefull)

Uma característica básica para a composição de serviços é a manutenção do estado transacional (ERL, 2010). A partir da manutenção de estado, é possível implementar tratamento de erros baseados em compensação e ter armazenamento operacional de todos as execuções do fluxo de composição.

Por definição, os serviços podem ser *statefull* (armazenamento de estado) ou *stateless* (não armazenamento de estado transacional) (ERL, 2010). Como principais limitadores e características de manutenção de estado transacional pode-se dizer que:

- Os serviços *stateless* não armazenam conteúdo das mensagens trafegadas, seu uso é simplificado quanto à transição e não permitem o uso de *roll-back* funcional (compensação);
- A operação de serviços *statefull* permite que o operador ou o analista tenha flexibilidade para analisar cada instância com falha, bem como o *payload* (conteúdo de mensagem) de cada operação contida no fluxo. Caso o operador queira analisar os tráfegos em serviços *stateless*, o único recurso disponível são os arquivos de log armazenados no servidor.
- O desempenho de serviços *stateless* é melhor quando comparado a serviços *statefull*, visto que não há armazenamento de estado, sendo apenas processamento pontual do fluxo.

Levando em consideração estas características, o orquestrador da ferramenta BPEL possui capacidade de manutenção de estado transacional (*statefull*).

3.2.3 Transformações e alterações de estruturas de dados

Tipicamente, em serviços compostos, há a necessidade de transformações de formato, tanto funcional quanto técnico, para a invocação de serviços componentes e destino de dados.

Do ponto de vista técnico, tem-se as seguintes transformações:

- Transformação do tipo de dados, que consiste na alteração de tipo de dados vindos da origem. Um exemplo típico é a transformação de formato *string* para data, ou *string* para inteiros;
- Transformação do formato de dados, que consiste na alteração da máscara de dados. Como exemplos tem-se o truncamento de dados *string*, alteração de formato de dados booleanos e alteração de estrutura de data.

Do ponto de vista funcional, tem-se as seguintes transformações:

- Transformação para enriquecimento de dados, que consiste de concatenações e atribuições de valores adicionais aos dados de origem, afim de atender à necessidade funcional de dados do sistema destino;

- Transformação de-para de dados, que é baseada em conversões de informações funcionais, em que os dados da origem são consultados em estruturas de dados de apoio (podendo ser tabelas ou serviços de-para) e transformados de acordo com destino. Tipicamente os alvos de transformaçãos de-para são as chaves/códigos de entidades. A tabela 1 apresenta um exemplo de estrutura de-para utilizada para transformações de chaves.

Tabela 1. Exemplo de estrutura de-para de dados

Chave- Origem	Chave- Destino
1	A
2	B
3	C

Dentro dos fluxos de serviços compostos, estas transformações são aplicadas em métodos *assign e transform*, porém utilizando diferentes sintaxes e extensões de acordo com as ferramentas.

Para o desenvolvimento utilizando o ferramental BPEL, tipicamente é utilizada a sintaxe de transformação xslt (*Extensible StyleSheet Language Transformation*), que, por definição, é uma linguagem para transformação de documentos XML e que utiliza as estruturas xpath para navegar entre as estruturas (W3C, 2012). A estrutura lógica da Figura 11 apresenta um exemplo de transformação de dados baseado em codificação xslt.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
```

```
      <xsl:for-each select="catalog/cd">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Figura 11 Exemplo de estrutura XSLT (W3C, 2012)

O exemplo apresentado (W3C, 2012) faz a atribuição de valores de título e artista para cada valor de cd dentro do catálogo, utilizando a estrutura *for-each*.

3.2.4 Tratamento de erros

Dadas as características apresentadas de transformações e invocações de serviços presentes nos fluxos compostos, os serviços componentes da composição podem apresentar falhas de execução, que podem ser de dois tipos: técnicas e funcionais de fluxo.

Em uma macro visão, as falhas técnicas podem ser mais fáceis de serem tratadas. As principais falhas técnicas são aquelas associadas a chamadas de serviços e falhas de transformações de dados:

- Falhas em chamadas de serviços são as falhas ligadas às invocações dos serviços. Como principais causas podem ser citados a indisponibilidade de serviços ou o não cumprimento do contrato de dados do serviço;
- Falha em transformação de dados são falhas ligadas às transformações e validações de novas estruturas. Podem estar vinculadas às falhas técnicas em fluxos BPEL.

Os tratamentos das falhas técnicas estão vinculados à arquitetura proposta para os serviços. Opções de tratamento técnico como retentativas ou notificação à origem

são válidas, visto que as falhas estão vinculadas à execução técnica, e não aos dados enviados pelos sistemas envolvidos.

As falhas funcionais estão tipicamente vinculadas às respostas dos serviços invocados. Um exemplo prático de falha funcional, é a invocação para cadastro de cliente que já seja existente no sistema destino. Neste caso, o sistema destino deve alertar que o cliente já está cadastrado e aplicar tratamento específico de falhas funcionais definidos como requisitos para o serviço. Exemplos de tratamentos de falhas funcionais são as notificações a sistemas envolvidos, armazenamento da mensagem para análise ou retentativa após um período definido.

Ainda como cenário de falha funcional, os serviços no fluxo de composição podem estar funcionalmente aninhados e a resposta ou falha de um serviço membro de composição pode exigir que haja um cancelamento de uma atividade executada. A atividade alternativa de cancelamento é denominado de compensação. Como exemplo de compensação, em um sistema de vendas, caso haja falha na criação da ordem de entrega, pode ser que seja necessário o procedimento de *roll-back* do pagamento já anteriormente invocado no fluxo do serviço composto de vendas.

Para solucionar estas possíveis falhas, a linguagem BPEL proporcia a adição de tratamento de erros de fluxo (*fault handlers*) permitindo a compensação/ *compensation* funcional do fluxo (IBM, 2012).

O desenvolvimento da composição considera a adição de um tratamento de falha ao escopo a ser compensado e, internamente ao tratamento de falha, efetua a adição do componente de compensação. Para a composição, define-se um novo fluxo alternativo, que tipicamente contempla uma correção funcional para os passo já executados com sucesso. No cenário de vendas anterior, uma alternativa seria adicionar no *compensation handler* de ordem de entrega um pagamento negativo ou simples cancelamento de pagamento.

3.2.5 Tarefas humanas (Human tasks)

Em serviços compostos, pode ocorrer a necessidade de uma atividade de aprovação manual ou de preenchimento manual de dados durante a execução do fluxo do serviço controlador.

Estas interações manuais são aplicadas por meio do componente BPEL chamado de *human task* (tarefa humana). Seu funcionamento consiste na pausa do

fluxo, podendo se fazer possível a desidratação da instância (ORACLE, 2012) e publicação do conteúdo do fluxo em formulários para análise do usuário.

Os formulários para preenchimento de dados devem possuir característica de aplicação e, portanto, devem apresentar usabilidade. Exemplos funcionais para uso de *human task* são aprovações de cadastros, vendas ou transações que não podem ser efetuadas sem intervenção e raciocínio humano.

3.3. Considerações do capítulo

Com base nas caraterísticas apresentadas, pode-se concluir que o desenvolvimento de composição está além da simples agregação de serviços.

Os serviços devem ser identificados e selecionados de acordo com regras de negócio e governança. Como características técnicas, os serviços compostos exigem a manutenção de estado e o tratamento analítico de interação entre serviços, erros funcionais, transformações e interação humana.

O desenvolvimento de composição não se mostra trivial e é importante que o ferramental disponível apresente recursos que permitam a aplicação das características técnicas ao desenvolvimento.

4. IDENTIFICAÇÃO DAS CARACTERÍSTICAS EM FERRAMENTA DE MERCADO

Dada a maturidade de soluções de mercado, através de um grande número de fornecedores e a complexidade de artefatos gerados, é fundamental o uso de uma ferramenta para desenvolvimento de uma solução SOA.

Foi, então, selecionada uma ferramenta do mercado para verificar os recursos que fornecessem apoio para incluir as principais características de um serviço composto, identificadas no capítulo 3, no produto desenvolvido com o seu uso.

4.1. Seleção de ferramenta de desenvolvimento

A ferramenta foi selecionada com base na análise apresentada no relatório de Gartner Group. Essa empresa realiza pesquisas, realização de programas, consultoria e eventos e tem, como principal objetivo, a introspecção necessária para tomadas de decisão quanto à seleção de serviços, ferramental e processos. (GARTNER, 2012).

Quanto à avaliação de ferramenta, a Gartner Group emite relatórios e gráficos qualitativos, através do método chamado de Quadrantes Mágicos (*Magic Quadrants*) (GARTNER, 2012). Esse método aplicado à avaliação de ferramentas SOA, resultou na Figura 12. Quadrante Mágico para SOA,que apresenta a avaliação dos principais fornecedores de ferramental SOA.

Figura 12. Quadrante Mágico para SOA (GARTNER, 2012)

Da análise do Quadrante Mágico do Gartner para SOA, a ferramenta selecionada para ser utilizada no trabalho é o pacote SOA da fornecedora Oracle. A ferramenta selecionada se coloca no Quadrante Mágico com mais alta posição no quesito de completude da visão e um bom posicionamento na habilidade de execução, além de possibilitar o *download* gratuito de toda ferramenta para projetos pessoais e acadêmicos.

O pacote *Oracle Fusion Middleware* é um conjunto de ferramentas para integrações de aplicação, baseados em orientação a serviços. Este pacote apresenta-se como uma solução madura, sendo composta de ferramental *Oracle Weblogic* (Servidor de Aplicação), *Oracle Service Bus*, *Oracle SOA Suite* (BPEL), *Oracle Data Integrator*, *Oracle Business Rules*, Oracle *Web Service Management System* (OWMS) (SOA SUITE, 2012).

Seguindo o indicador apontado pelo Gartner, para o desenvolvimento específico das características e análise baseados em Orquestração, foi utilizado a ferramenta Oracle SOA Suite.

Todas as características técnicas de desenvolvimento SOA apresentadas no capítulo 3.3 são identificadas e pontualmente implentadas para avaliar aderência quanto as citações acadêmicas.

4.2. Identificação das características de serviços compostos na ferramenta

Na ferramenta Oracle SOA Suite, o primeiro objeto a ser criado é o composto, nomeado de *composite.xml*. Nele são criados os fluxos BPEL, *Human task* e demais componentes de controle de mensagens, fornecidos pela ferramenta. Ao criar o componente BPEL no composite.xml, a ferramenta apresenta a tela vazia, onde o fluxo vai ser desenvolvido conforme a figura 13.

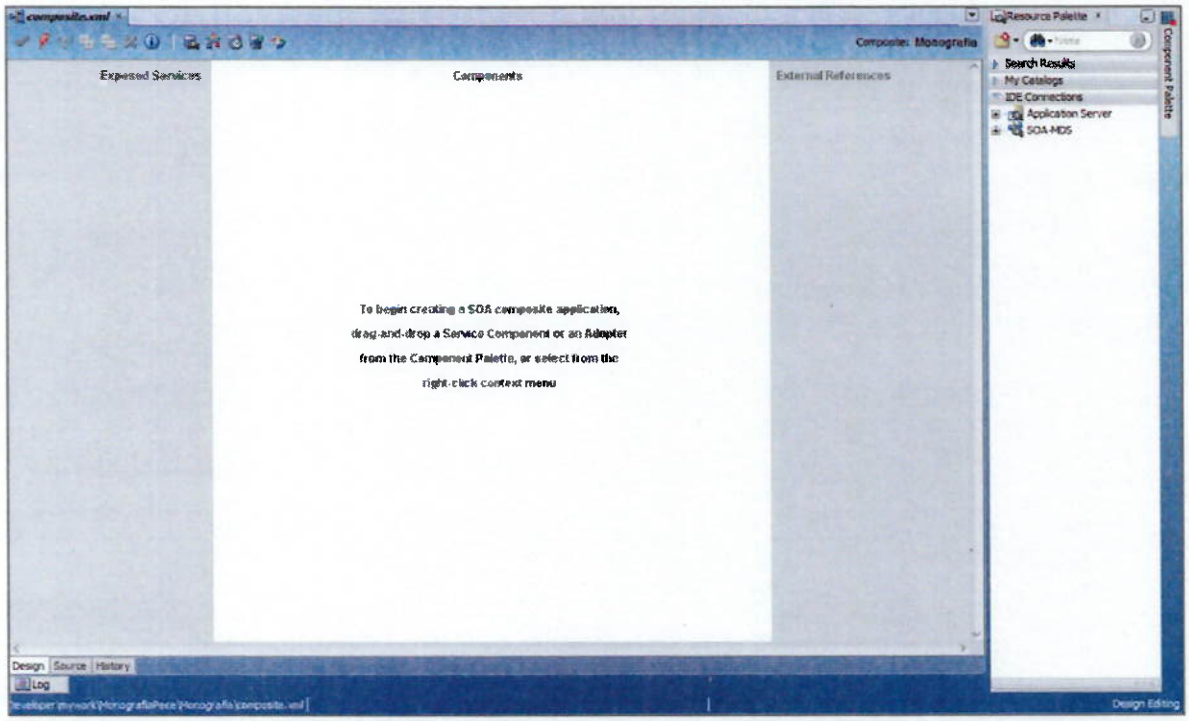

Figura 13. Visualização de Composite vazio no Oracle Soa Suite

4.2.1 Identificação da característica de invocações síncronas, assíncronas e fire-forget

Após a criação do componente *composite.xml*, pode-se inserir o fluxo BPEL, *human tasks* e demais componentes de controle de mensagens. Na criação do fluxo BPEL, a ferramenta solicita o tipo invocação da mensagen do serviço BPEL, aplicando a característica de sincronismo, assíncronismo e *fire-forget* (*one-way na ferramenta*). Como exemplo, na figura 14, é criado um serviço assíncrono.

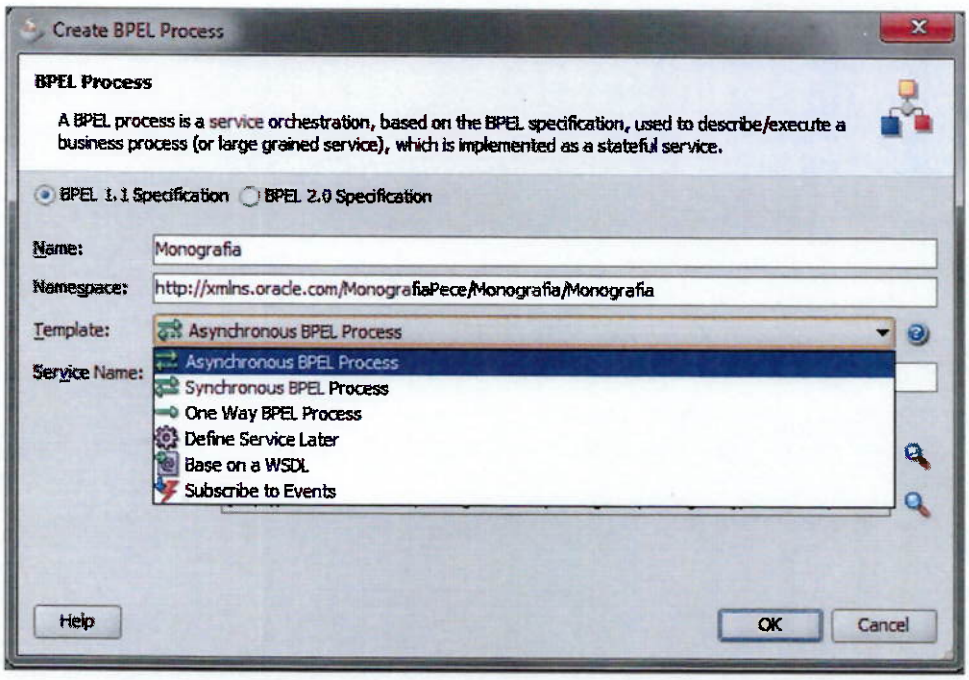

Figura 14. Tipo de Invocação BPEL

Após a criação do componente *composite.xml*, a ferramenta habilita a edição do fluxo, seguindo as diretrizes da linguagem BPEL (figura 15). É apresentado um menu de opções (figura 16) para criação de componentes BPEL elicitados durante a seção 2.2.3 e de adaptadores de dados e de componentes nativos da solução SOA Suite.

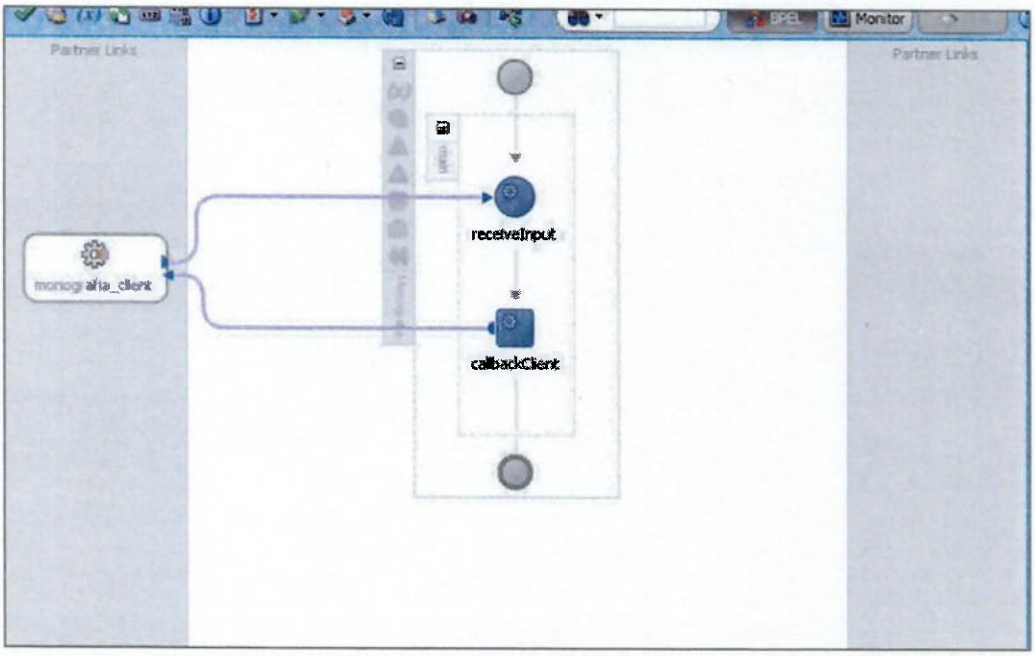

Figura 15. Fluxo BPEL automático

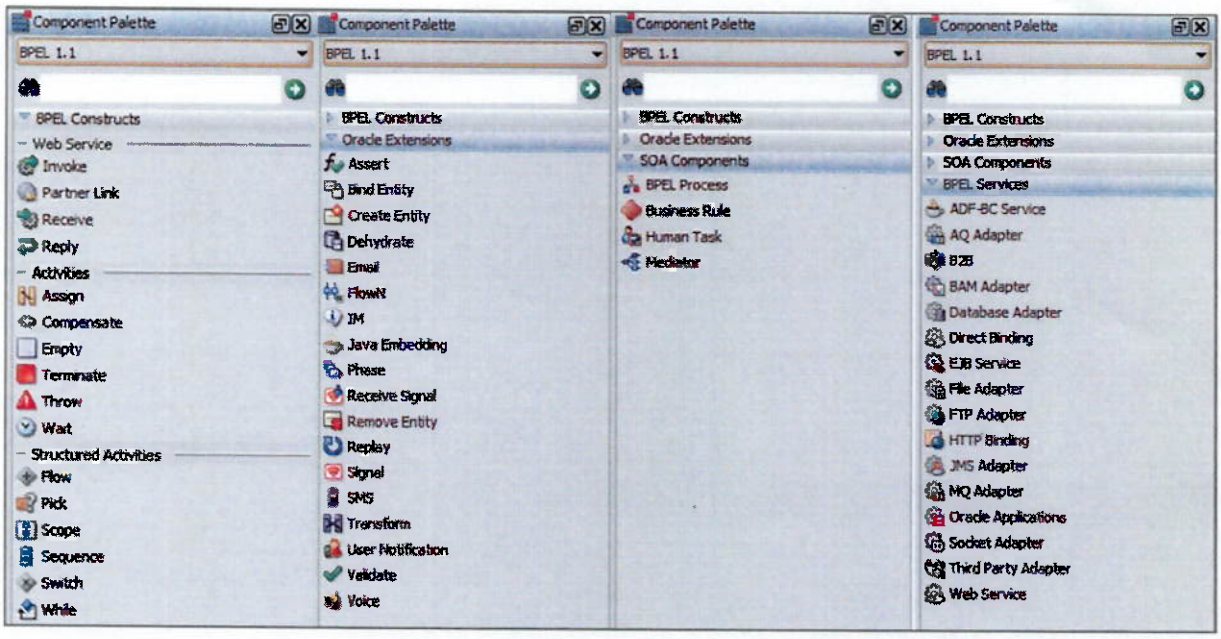

Figura 16. Componentes *Pallet* BPEL Oracle

4.2.2 Identificação da característica de transformação de dados

A partir do fluxo BPEL e da barra de desenvolvimento, é criado um processo de transformação e chamada a um serviço externo. Para a transformação de *lay-out* de dados é utilizado o componente *transform*, que faz uso de uma estrutura xslt (como mencionado na seção 3.2.3).

O componente *transform (figura 17)* permite a edição do *lay-out* de dados xml a partir de dados de entrada e uma variável destino. O modelo de transformação, ou seja, o código a ser aplicado se encontra em um arquivo com extensão xsl que é criado de forma automática pela ferramenta (figura 17). A figura 18 aponta o editor de parâmetros de entrada para a criação das transformações xsl de estruturas de mensagens.

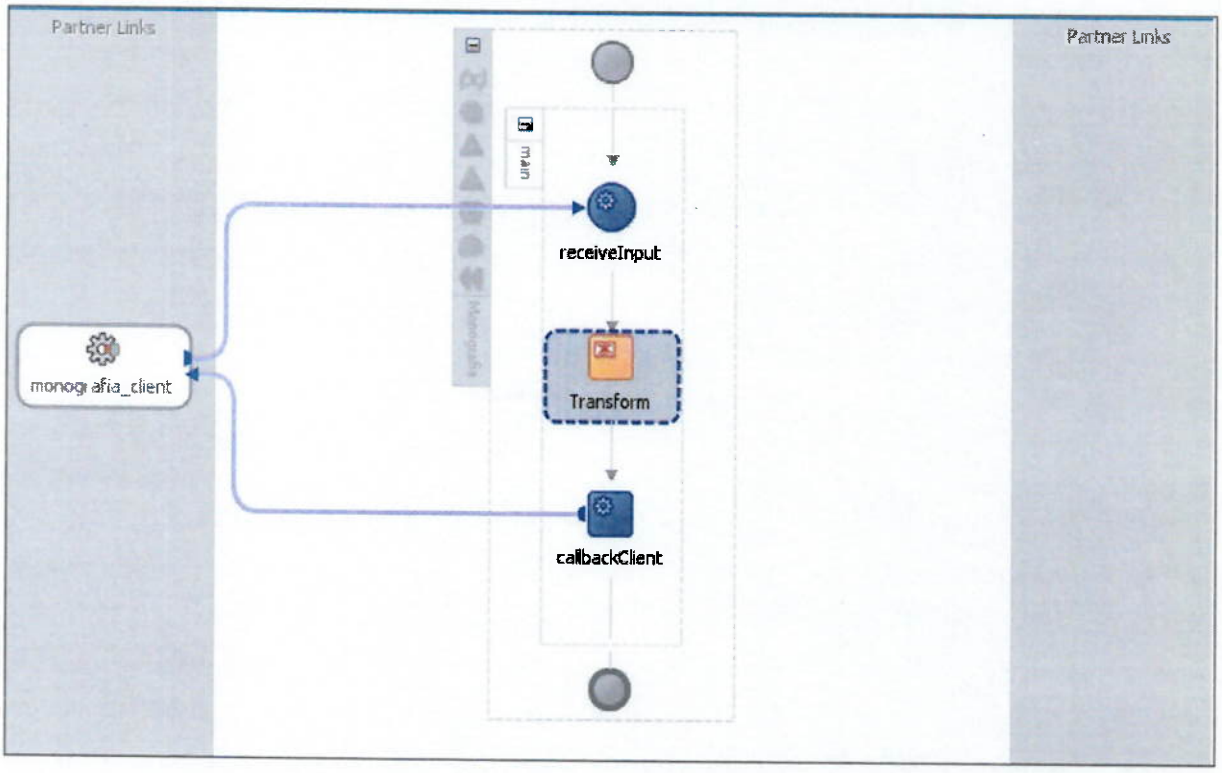

Figura 17. Desenvolvimento componente *Transform*

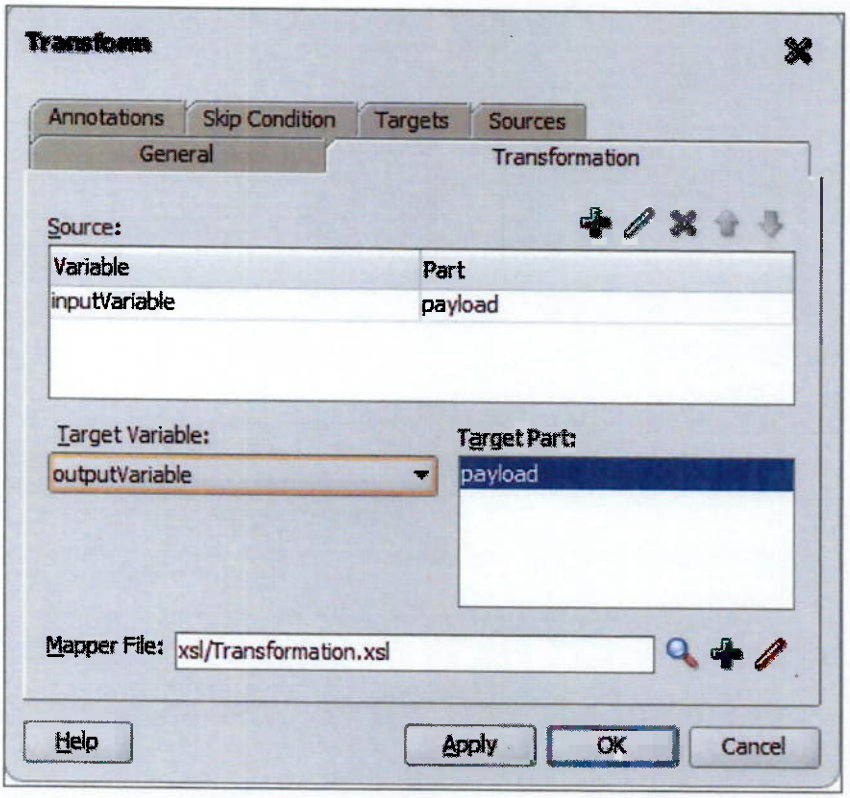

Figura 18. Definição de Parâmetros para *Transform*

4.2.3 Identificação da característica de tratamento de erros

Como mencionado na seção 3.2.4, outra característica fundamental para o desenvolvimento de fluxo de composição é o tratamento de erros.

Os tratamentos de erros técnicos são realizados com base no componente *catch*, que é responsável por capturar a falha durante o fluxo e executar um tratamento técnico, como mencionado na sessão 3.3.9.

No exemplo da figura 19, caso haja um erro técnica durante a transformação de dados o componente *catch* captura a falha durante a transformação e emite um e-mail através do componente *e-mail*.

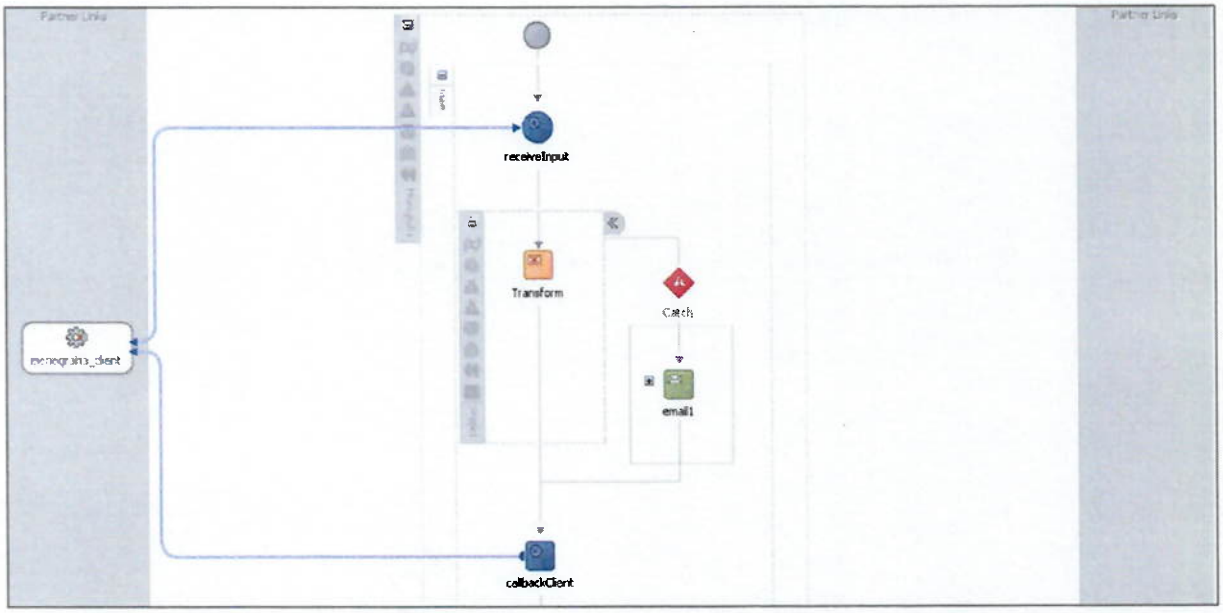

Figura 19. Uso do componente *catch*

Em caso de erro funcional, conforme mencionado na seçaõ 3.2.4, o fluxo BPEL deve usar o componente *compensate*, que faz o controle do fluxo transacional *statefull* do fluxo BPEL e dispara a execução de um fluxo alternativo correspondente ao tratamento da falha funcional encontrada, que inclui um componente compensation, conforme apresentado na figura 20.

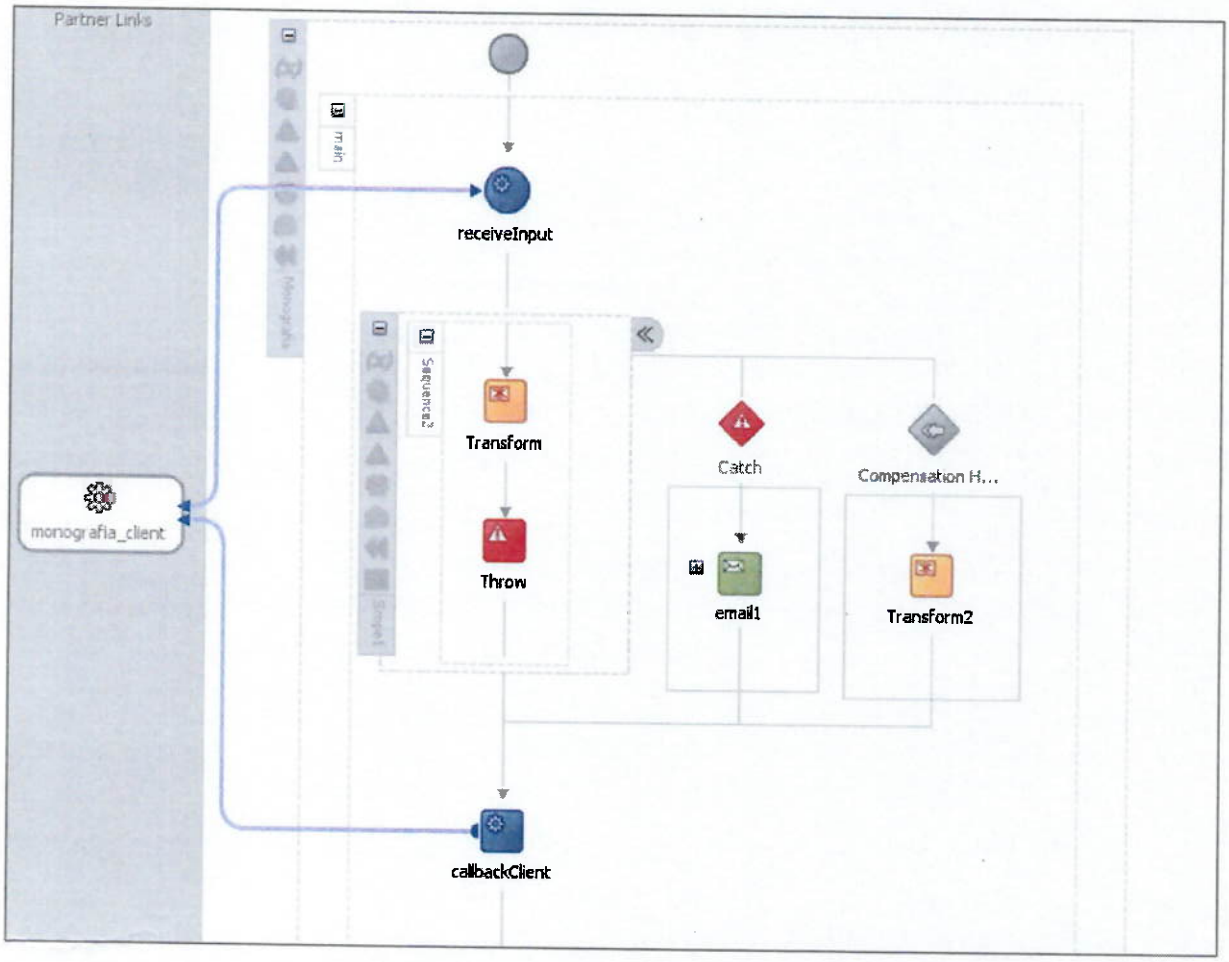

Figura 20. Uso do componente *compensation*

4.2.4 Identificação da característica de Human task

A característica de *Human task* está presente no fluxo BPEL para representar as atividades de aprovação e intervenção durante a execução.

Apresentadas na seção 3.2.5, as *Human tasks* são introduzidas no fluxo técnico através do componente *human task*. Este componente cria uma camada de aplicação com interface de usuário, a qual interage com o fluxo BPEL e o componete resultante deve ser acoplado ao componente *composite.xml*, conforme ilustrado na figura 21.

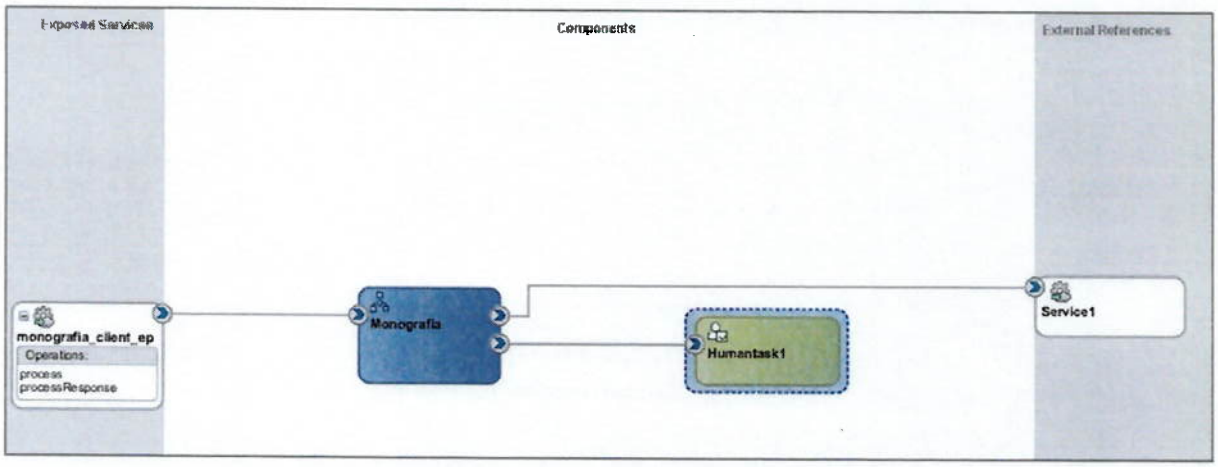

Figura 21. Adição de *Human task* no *composite*.xml

Com a inserção de *human task*, esse componente cria um fluxo com três possibilidades para a resposta do usuário: usuário rejeita o dado do fluxo, usuário aprova o dado do fluxo ou usuário repassa para uma solução alternativa (para alternativas não booleanas) (figura 22). Durante a execução, o usuário recebe um formulário de aprovação de dados, como apresentado na figura 23.

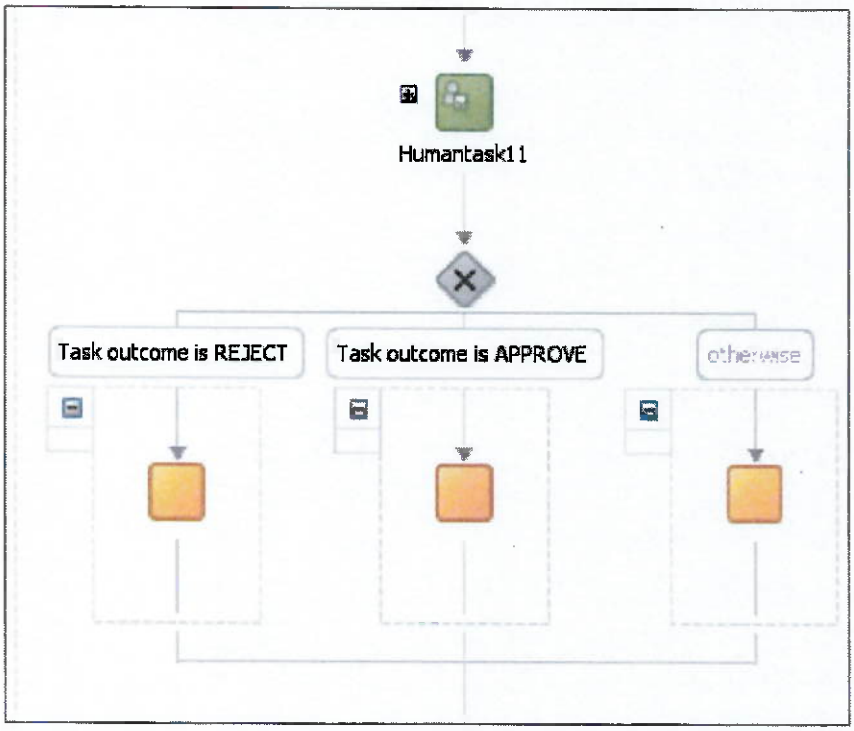

Figura 22. *Human task* no fluxo BPEL

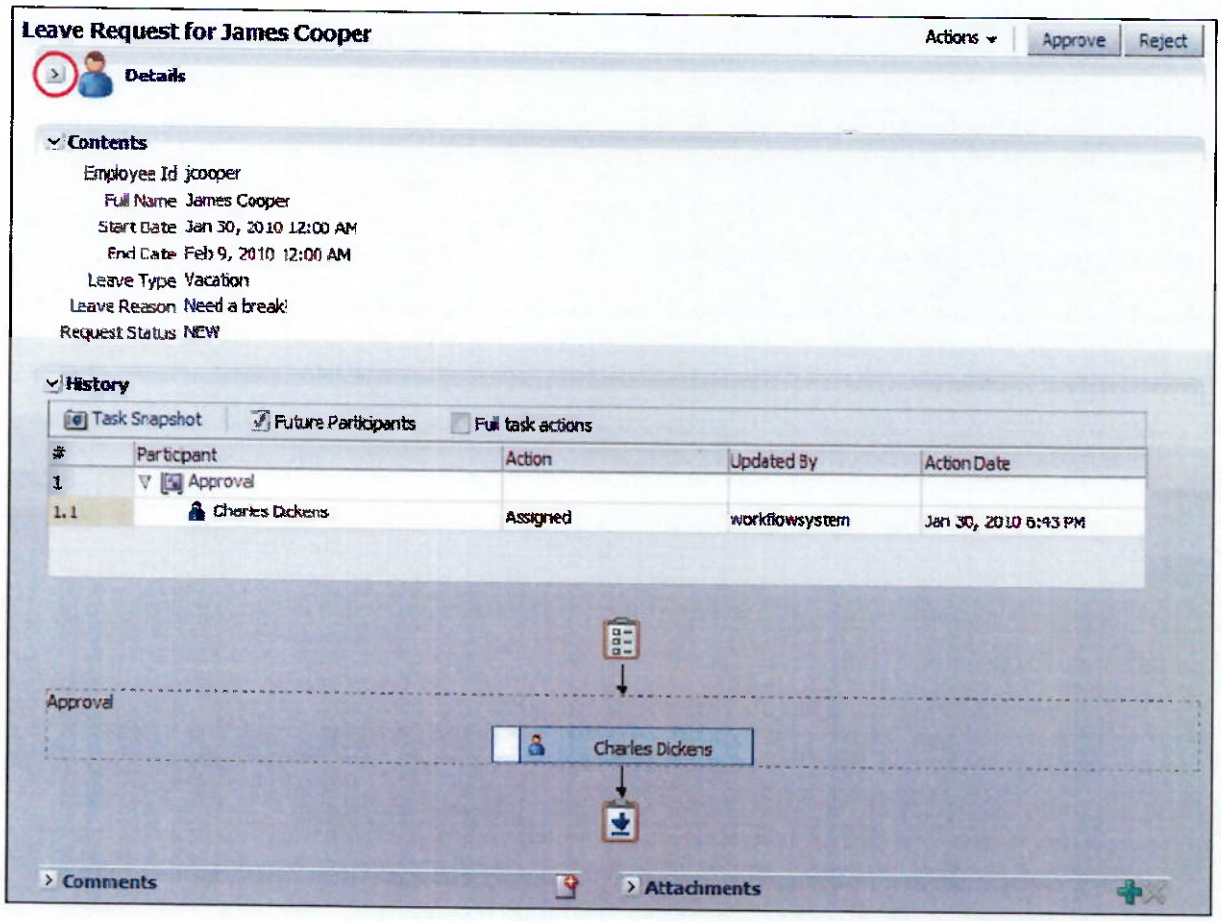

Figura 23. Formulário JSP criado automaticamente por *human task*

A figura 23 apresenta a aplicação de um formulário *Human task* aplicado na tecnologia JSP. No formulário o usuário "Charles Dickens" aprova um pedido de férias.

4.2.5 Identificação da característica de manutenção de estado (statefull)

A característica de manutenção de estado transacional pode ser notada durante a execução do fluxo BPEL.

Todos os fluxos BPEL são implantados em servidores de aplicação que fazem o controle de execução e operação dos componentes desenvolvidos. Para cada fluxo é possível recuperar as mensagens trafegadas e nota-se o armazenamento persistente dos dados. Esta persistência permite a visualização de *pay-load* XML e possibilita o tratamento de compensação transacional.

No pacote Oracle SOA Suite, os fluxos são implantados no servidor de aplicação Weblogic e a visualização e gestão dos serviços BPEL é feita pela ferramenta/portal Oracle Enterprise Manager, como apresentado na figura 24.

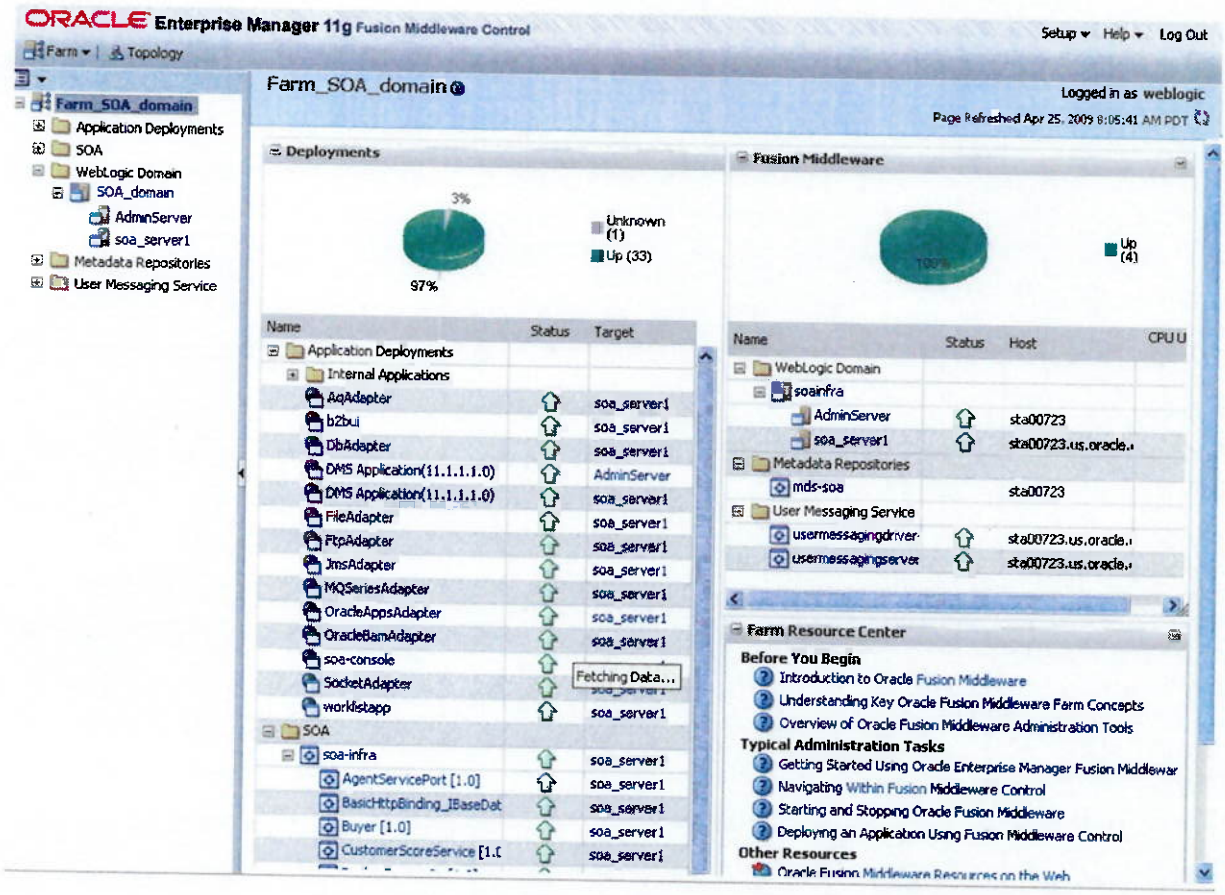

Figura 24. Visão Geral Oracle Enterprise Manager

A figura 24 apresenta a console de adminitração do *Oracle Enterprise Manager*, na coluna esquerda são apresentados as instâncias de servidores onde os serviços são implantados. Na coluna central (*Deployments*), todos os serviços e adaptadores de dados implantados são apresentados, bem como sua disponibilidade atual.

4.3. Considerações do capítulo

Com base neste capítulo, é possível concluir que as características identificadas no capítulo 3 estão presentes na ferramenta analisada e constituem

uma parte fundamental para o desenvolvimento de fluxos de composição na ferramenta *Oracle SOA Suite*.

As características de invocações de serviços, transformações, tratamento de erros e *human tasks* podem ser identificadas durante o desenvolvimento. Para a conceituação de manutenção de estado, é necessário executar o fluxo no servidor e verificar que os dados trafegados e os resultados da execução do fluxo são armazenados de forma persistente.

A ferramenta *Oracle SOA Suite* ainda apresenta grande variação de cenários, adaptadores técnicos e linguagens próprias, porém as principais características apontadas formam a base para o desenvolvimento BPEL levando em prática as referências acadêmicas.

5. CONSIDERAÇÕES FINAIS

Este capítulo tem a finalidade de apresentar as conclusões relacionadas com as características de composição apresentadas no capítulo 3 e o suporte da ferramenta analisado no capítulo 4. Além disso, apresenta as contribuições alcançadas e trabalhos futuros deste trabalho.

5.1. Conclusões

Com base nos referenciais teóricos, nas características de serviços compostos apresentadas e no resultado da análise da ferramenta de mercado selecionada, é possível concluir que, durante o processo de análise e desenvolvimento de serviços compostos, há fatores de alta complexidade que impactam na elaboração de composição de serviços e que devem ser levados em consideração.

Tipicamente tem-se a falsa ideia de que o desenvolvimento de serviços compostos é simples, uma vez que parte dos serviços membros de composição pode estar desenvolvida. Todavia, as características apresentadas contradizem este raciocínio e apresentam que, além de domínio técnico do fluxo a ser desenvolvido, a composição exige o conhecimento funcional do fluxo.

A ferramenta analisada permite o desenvolvimento de serviços compostos baseado nas características apresentadas o que leva a observar que existe alinhamento da ferramenta selecionada do mercado em relação às práticas de desenvolvimento de serviços compostos SOA estudadas em referêncial acadêmico.

Portanto, é possível constatar que a ferramenta de mercado apontada como referência de mercado está alinhada com as referências acadêmicas, e que o desenvolvimento de serviços compostos está além de simples codificação e agregação de serviços. Conclui-se que o desenvolvimento de composição está fortemente atrelado ao domínio funcional de negócio e as características técnicas do fluxo e dos serviços envolvidos.

5.2. Contribuições do Trabalho

Como contribuição têm-se a apresentação e agrupamento das principais características de desenvolvimento de serviços compostos SOA e a análise destas características em uma ferramenta de mercado.

De maneira geral, a literatura existente para composição não aborda de forma agrupada as principais características e não apresenta desenvolvimento prático, deixando então uma lacuna, que foi considerada como oportunidade para colaboração acadêmica e foi utilizada neste trabalho.

As características apresentadas em ferramenta de mercado possibilitam que desenvolvedores e analistas de seviços compreendam, através dos seus recursos, as principais complexidades e preocupações existentes durante o processo de desenvolvimento de serviços compostos.

5.3. Trabalhos futuros

A partir do trabalho apresentado, é possível evoluir a análise de composição para outras abordagens de análise, como a *botton-up*, e atingir características complementares às apresentadas.

Como possibilidades futuras, pode-se aplicar o estudo sobre desenvolvimento de serviços compostos BPEL de forma integrada a ferramentas de desenvolvimento BPM.

Também é possível encarar, como oportunidade, o estudo de manuntenção de serviços compostos, abordando processos de manutenção evolutiva do serviços e evolução dos serviços membro de composição.

Quanto à análise e à definição de serviços, é possível definir trabalho associando os serviços compostos ao conceito de modelagem canônica de serviços, em que todos contratos de dados e operações são geridos de forma centralizada por áreas de Arquitetura Corporativa, possibilitando maior integração entre áreas de negócio.

6. REFERÊNCIAS

1. BENETT, S.G; GEE, C.; MANES A. T; SCHNEIDER,R.; ERL; T.; SOA Governance; Boston EUA: Prenctice Hall, 2009, 2º ed.

2. BIEBERSTEIN, N., BOSE, S., FIAMMANTE, M., JONES, K., SHAH, R.: Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap. Boston: Prentice-Hall, Englewood Cliffs, 2005, 1º ed.

3. DIMAGGIO,L.; CONNER, K.; MAGESH, K.; CUNNINGHAM,T.; Jboss ESB – beginner's guide, Florida: Packt Publishing, 2012, 1º ed.

4. ERL, T.; Principles of Service Design, Boston EUA: Prenctice Hall, 2010, 2º ed.

5. ERL, T.; SOA Design Patterns, Boston EUA: Prentice Hall, 2008, 2º ed.

6. FAREGHZADEH, N. Service Identification Approach to SOA Development. World Academy of Science, Engineering and Technology, 2010.

7. GARTNER. Disponível em http://www.gartner.com/technology/about.jsp. Acessado em: 08 nov. 2012.

8. IBM WS BPEL. Disponível em http://www.ibm.com/developerworks/webservices/library/ws-bpelcol6. Acessado em: 11 ago. 2012.

9. OMG; Business Process Model and Notation (BPMN). Disponível em http://www.omg.org/spec/BPMN/2.0. Acessado em: 25 set. 2012.

10. ORACLE. Disponível em http://www.oracle.com/technetwork/articles/soa/wli-bpel-transactions-088255.html. Acessado em: 11 ago 2012.

11. Oracle[2] SOA Blueprint. Disponível em: http://www.oracle.com/technetwork/articles/entarch/blueprint-soa-integration.087583.html. Acessado em 23 jan. 2013

12. Oracle[3] SOA Suite. Disponível em http://www.oracle.com/br/products/middleware/overview/index.html. Acessado em: 25 jan. 2013.

13. PLESUMS, C.A.; Introduction to Workflow. Disponível em http://www.plesums.com/image/introworkflow.html – Acessado em: 26 out 2012.

14. REYNOLDS, A.; WRIGHT, M; Oracle Soa Suite 11g R1 Developers Guide, Packt Publishing, 1º ed., 2010.

15. SOA Working Group of The Open Group. Disponível em https://collaboration.opengroup.org/projects/soa/ - Acessado em: 14 set de 2012.

16. W3C Schools. Disponível em http://www.w3schools.com/xsl. Acessado em 23 out. 2012

17. What's reusable Service – Disponível em: http://www.oracle.com/technetwork/articles/bechara-reusable-service-087796.html. Acessado 23 ago 2012.

18. YANAI, L.P.R; MISLP: Método de Identificação de Serviços Baseado em Linguagens de Padrões, São Paulo, 2010, 125 p.