BRUNO LINS DE OLIVEIRA

PROPOSTA DE PROCESSO DE CRIAÇÃO DE LINGUAGEM DE DOMÍNIO ESPECÍFICO (DSL) EM APLICAÇÕES COM ARQUITETURA ORIENTADA A SERVIÇO

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
2011

BRUNO LINS DE OLIVEIRA

PROPOSTA DE PROCESSO DE CRIAÇÃO DE LINGUAGEM DE DOMÍNIO ESPECÍFICO (DSL) EM APLICAÇÕES COM ARQUITETURA ORIENTADA A SERVIÇO

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: Prof. Dr. Kechi Hirama

São Paulo

2011

FICHA CATALOGRÁFICA

Oliveira, Bruno Lins de
Proposta de processo de criação de linguagem de domínio específico (DSL) em aplicações com arquitetura orientada a serviço / B.L. de Oliveira. -- São Paulo, 2011.
70 p.

Monografia (MBA em Tecnologia de Software) – Escola Politécnica da Universidade de São Paulo. Programa de Educação Continuada em Engenharia.

1. Arquitetura de software 2. Engenharia de software 3. Processo de software 4. DSL 5. SOA I. Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II. t.

DEDICATÓRIA

Dedico este trabalho a minha noiva Natalye, por estar ao meu lado durante todo período de elaboração e aos meus pais que são os responsáveis por tornar tudo isso possível.

AGRADECIMENTOS

À Escola Politécnica da Universidade de São Paulo – EPUSP que incentiva o crescimento intelectual dos alunos disponibilizando inúmeros recursos.

Ao PECE – Programa de Educação Continuada em Engenharia por ter me dado a oportunidade de cursar este MBA.

Ao meu orientador Prof. Dr. Kechi Hirama, por todo o conhecimento e experiência passados durante o processo de elaboração deste trabalho.

Aos meus pais e noiva que contribuíram, mesmo que de forma indireta, para a realização deste trabalho.

Por mais longa que seja a caminhada o mais importante é dar o primeiro passo.

Vinícius de Moraes

RESUMO

Em um cenário em que aplicações distribuídas se tornam cada vez mais comuns, o padrão arquitetural SOA surge como base para organização das mesmas. Entretanto, devido à necessidade da arquitetura estar aderente ao processo de negócio, é necessário que os profissionais de negócio sejam capazes de especificar, modelar e entender os serviços desta arquitetura. Com isso, o presente trabalho tem como objetivo propor um processo de criação de uma linguagem específica de domínio para SOA. Para isso, são definidos os conceitos de DSL e SOA, apresentados trabalhos relacionados e as etapas do processo proposto, além de realizar a aplicação do mesmo em um projeto real. Esta aplicação irá avaliar se o processo de criação DSL é eficaz para uso efetivo em projetos com arquitetura SOA.

Palavras-chave: SOA, DSL, processo de software, arquitetura de software

ABSTRACT

In a scenario in which distributed applications are becoming more common, the SOA architectural pattern emerges as a base to organize them. However, due to the need of the architecture to be adherent to the business process, it is necessary that business professionals be able to specify, model and understand the services. Therefore, this paper aims to propose a process of creating a domain-specific language for SOA. To do this, it will be defined the concepts of DSL and SOA, presenting related papers and steps of the proposed process, besides performing its application on a real project. This application will assess whether the process of creating DSL is efficacious for effective use in projects with SOA architecture.

Keywords: SOA, DSL, software process, software architecture.

LISTA DE ILUSTRAÇÕES

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
DSL	Domain-Specific language (Linguagem de domínio específico)
SOA	Service-oriented architecture (Arquitetura orientada a serviços)
SOAP	Simple Object Access Protocol
WSDL	Web Services Description Language
WS-BPEL	Web Services Business Process Execution Language
XSD	XML Schema Definition

SUMÁRIO

Pág.

1. INTRODUÇÃO

Este capítulo apresenta motivações, objetivo e justificativas que levaram ao desenvolvimento desse trabalho, além de descrever como o mesmo é estruturado.

1.1 Motivações

Devido ao cenário econômico atual, em que transações entre empresas em países distintos e a utilização de sistemas integrados para garantir a eficiência dos processos da empresa se tornaram cada vez mais comuns, o uso da tecnologia se tornou imprescindível com a Internet surgindo como a principal delas.

Este cenário exige flexibilidade e capacidade de evolução dos negócios da empresa, e cada vez mais para garantir isso, vem se utilizando o padrão arquitetural que define o Serviço como elemento principal, denominado SOA (Arquitetura Orientada a Serviços), que é descrito como um paradigma para organização e utilização de competências distribuídas que estão sob controle de diferentes domínios proprietários, segundo o modelo de referência (OASIS OPEN, 2006).

Este padrão arquitetural permite a correlação entre as necessidades de negócio de uma determinada empresa, com as capacidades expostas pelos serviços da própria empresa ou de terceiros, permitindo muitas vezes redução de custos.

Apesar de sua definição, SOA muitas vezes é apresentado com uma solução estritamente tecnológica, porém a mudança de uma empresa para uma arquitetura orientada a serviços significa a publicação, externamente ou internamente, de seus processos de negócios. Desta forma, a tecnologia serve apenas com base para manter a solução disponível.

Analisando estes aspectos é possível notar que para o desenvolvimento de soluções com esse padrão arquitetural, as transformações dos modelos definidos pelos especialistas de negócio em linguagem executável se tornam um aspecto primordial para o sucesso do projeto.

Com a realização dessas transformações, diminuiriam os problemas de comunicação entre a equipe de negócio e técnica, permitindo que os especialistas

de negócio realizem suas atividades de modelagem do negócio enquanto a equipe técnica forneceria a infraestrutura para execução e manutenção da solução.

Com isso, é necessário que especialistas de negócio tenham a capacidade de especificar, modelar e entender os serviços utilizando linguagens de domínio do negócio. Além disso, essas linguagens permitiriam a interoperabilidade entre os modelos de negócio e a linguagem na qual ele será implementada, tal como WS-BPEL, caso isto não ocorra poderia levar ao abandono dessa prática. (CARDENAS, LIU, *et al.*, 2010)

A utilização dessa estratégia de modelagem dos serviços garantiria a facilidade de evolução dos serviços, baseando-se nas mudanças ocorridas no processo de negócio.

1.2 Objetivo

O objetivo deste trabalho é propor um processo de criação de uma Linguagem de Domínio Específico (DSL) em soluções de arquitetura orientada a serviços com o objetivo de detectar benefícios e problemas associados a esta abordagem.

Esta análise é realizada através da comparação dos dados obtidos da aplicação da Proposta de Processo de Criação da DSL, apresentado neste trabalho, com os dados obtidos através da execução do mesmo, utilizando outras técnicas. Além dessa comparação, é apresentada a análise de casos relacionados.

1.3 Justificativas

O desenvolvimento utilizando DSL é uma prática que já vem sendo discutida e utilizada desde a década de 80 (PIETRO-DIAZ, 1987), porém apenas um número limitado de ferramentas de apoio estava disponível no mercado, o que inviabilizava a adoção dessa prática. Nos últimos anos grandes fabricantes de softwares iniciaram o processo de criação dessas ferramentas o que vem tornando a adoção desta técnica novamente viável.

Mesmo com os possíveis benefícios do uso dessas linguagens, ainda não existem muitos estudos relacionados a essa técnica (OBERORTNER, ZDUN e DUSTDAR, 2008), o que torna esta pesquisa um trabalho exploratório com o objetivo de contribuir para apresentar aspectos para futuras pesquisas acadêmicas.

Além disso, através da aplicação apresentada, este trabalho apresenta algo que poderá ser aplicado em projetos reais, assim como os casos apresentados em trabalhos relacionados.

A aplicação da proposta apresentada no desenvolvimento de soluções usando o conceito SOA, tem como resultado uma aplicação desenvolvida em uma linguagem de alto nível (negócio) que pode ser executada através de ferramentas que permitam o gerenciamento e execução de WS-BPEL ou outras linguagens de execução. Como exemplo de ferramentas que possuem essa capacidade, podem-se citar os *middlewares*, ORACLE BPEL Process Manager, IBM WebSphere, Microsoft BizTalk Server, Windows Server AppFabric.

1.4 Estrutura do Trabalho

O Capítulo 1 INTRODUÇÃO apresenta as motivações, o objetivo, as justificativas e a estrutura do trabalho.

No Capítulo 2 REVISÃO BIBLIOGRÁFICA é descrito o conceito SOA, realizada a caracterização de DSL e exemplifica algumas que podem ser utilizadas para uma solução SOA e apresentará, em detalhe, todas as tecnologias que serão utilizadas no desenvolvimento do experimento.

O Capítulo 3 LINGUAGENS DE DOMÍNIO ESPECÍFICO ORIENTADAS A SERVIÇOS apresenta trabalhos relacionados que foram utilizados como ponto de partida para a proposta.

O Capítulo 4 PROPOSTA DE PROCESSO DE CRIAÇÃO DA DSL propõe um conjunto de atividades necessárias para a criação de uma DSL utilizando fluxos de trabalhos.

No Capítulo 5 APLICAÇÃO DA PROPOSTA é apresentado o desenvolvimento de uma solução de Aprovação de Crédito que possui arquitetura orientada a serviços, onde é realizada a utilização de uma linguagem de domínio juntamente com as tecnologias apresentadas neste trabalho. Esta aplicação tem

como objetivo avaliar a proposta de processo descrita neste trabalho, além de descrever os benefícios e problemas encontrados durante o processo de desenvolvimento.

O capítulo 6 CONSIDERAÇÕES FINAIS descreve as conclusões e as contribuições do trabalho e sugestões de trabalhos futuros que podem ser desenvolvidos a partir deste trabalho.

A seção REFERÊNCIAS apresentada as referências utilizadas para a elaboração deste trabalho.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo é apresentada a definição de todas as tecnologias utilizadas neste trabalho, baseando-se nas informações obtidas nas literaturas relacionadas.

2.1 Linguagens de domínio específico (DSL)

É comum no desenvolvimento de software o uso de linguagens como C# ou Java e essas linguagens são definidas como “linguagens de propósito geral”, ou seja, atendem a todos os domínios de maneira sucinta. Já linguagens que têm como objetivo um único tipo de problema são chamadas linguagens de domínio específico ou simplesmente DSL.

DSL são linguagens de programação de expressividade limitada focada num domínio particular. (FOWLER, 2008)

Essas linguagens têm como principal função, expressar em alto nível de abstração, aspectos específicos de um domínio de negócio usando conceitos do mesmo. A utilização dessas linguagens simplifica e garante confiabilidade no processo de implementação. (ROSA, AMARAL e BARROCA, 2009)

Muitas dessas linguagens fazem parte do dia-a-dia de um desenvolvedor de software há muitos anos, porém não são normalmente definidas como DSL. Entre as linguagens mais utilizadas podemos citar:

- HTML: Linguagem usada para produzir páginas Web.
- CSS: Linguagem de estilo usada para definir a apresentação de documentos escritos com HTML ou XML.
- SQL: Linguagem de consulta em banco de dados relacional.
- MatLab: Linguagem de alto nível que permite a execução de tarefas computacionais.

Além disso, é possível realizar a criação de linguagens específicas para o domínio de uma empresa ou projeto, exemplos desse processo são apresentados no capitulo 4 através da descrição de estudos relacionados.

Segundo Martin Fowler (FOWLER, 2005), pode-se categorizar essas linguagens em – DSL Internas e DSL Externas.

2.1.1 DSL Externas

Toda DSL criada através de uma linguagem diferente da linguagem principal de uma aplicação é denominada DSL Externa. Na Figura 1, pode-se ver um exemplo de arquivo de configuração XML que representa o mapeamento de um arquivo de texto para um domínio específico.

```
<ReaderConfiguration>
  <Mapping Code = "SVCL" TargetClass = "dsl.ServiceCall">
    <Field name = "CustomerName" start = "4" end = "18"/>
    <Field name = "CustomerID" start = "19" end = "23"/>
    <Field name = "CallTypeCode" start = "24" end = "27"/>
    <Field name = "DateOfCallString" start = "28" end = "35"/>
  </Mapping>
  <Mapping Code = "USGE" TargetClass = "dsl.Usage">
    <Field name = "CustomerID" start = "4" end = "8"/>
    <Field name = "CustomerName" start = "9" end = "22"/>
    <Field name = "Cycle" start = "30" end = "30"/>
    <Field name = "ReadDate" start = "31" end = "36"/>
  </Mapping>
</ReaderConfiguration>
```

Figura 1 - Exemplo de arquivo de configuração XML (FOWLER, 2005).

Devido ao fato de não estar limitada a uma linguagem base, a definição dessas linguagens depende apenas da habilidade do analista na sua criação e isso possibilita expressar o domínio de forma mais fácil. Por outro lado, estas DSL necessitam de um tradutor que permita realizar a sua interpretação. Dependendo da complexidade da DSL, o tradutor pode tornar-se um grande impedimento para o uso de uma DSL.

Além do problema citado, a falta de integração entre a linguagem base e a DSL afeta o uso de recursos disponibilizados por IDE atuais, como refatoração automática.

Existem muitas interpretações equivocadas em relação à aplicabilidade de uma DSL Externa, que indicam que ao criar uma nova linguagem estariam aumentando a complexidade do projeto, pois haveria mais linguagens para serem

interpretadas. Ao contrário desta interpretação, as DSL são simples e limitadas ao domínio, o que as torna de fácil aprendizado.

2.1.2 DSL Internas

Diferente das DSL Externas apresentadas anteriormente, as DSL Internas aplicam todos os recursos de uma linguagem base e de suas ferramentas. Com isso, não é mais necessário a criação de interpretadores citados anteriormente (FOWLER, 2005). O *Ruby on Rails*, linguagem específica para o desenvolvimento de aplicações WEB baseada no Ruby, é uma das DSL Internas mais usadas na atualidade. A Figura 2 apresenta um exemplo de código que realiza o controle de páginas de gerenciamento de usuários, criado a partir da linguagem *Ruby on Rails*.

Figura 2 - Exemplo de código Ruby on Rails (RAILS, 2010).

Como ocorre na linguagem apresentada acima, as DSL Internas têm como propósito incluir, em uma linguagem existente, funções específicas de um determinado domínio.

Um dos pontos fortes que levam o uso de uma DSL é o fato de possibilitar o desenvolvimento de um software sem o total conhecimento de uma linguagem, porém pode ser necessário o conhecimento da linguagem base para o uso de DSL Internas, o que pode se tornar um possível impedimento.

2.1.3 Workbench de Linguagem

É de conhecimento de toda a área de desenvolvimento de software que um dos grandes causadores do insucesso de projetos é a falta ou falha de comunicação entre os participantes.

Com isso, é comum uma proposta de desenvolvimento usando DSL estar atrelada à ideia de que solucionariam os problemas nesta área, porém a simples inclusão de uma DSL no ciclo desenvolvimento apresenta grandes problemas, como mostrados anteriormente, que dificultam o envolvimento do profissional responsável pela modelagem de negócio no processo de desenvolvimento, principalmente devido ao fato de apresentarem a necessidade de conhecimento tecnológico.

Em virtude disso, Martin Fowler (FOWLER, 2005) apresentou o termo Workbench de Linguagem, que descreve uma nova classe de ferramentas de desenvolvimento que permitem o desenvolvimento com DSL de forma integrada. Essas ferramentas apresentam as seguintes características:

- Possibilidade de definição de um Modelo Semântico para a linguagem;
- Definição de um ambiente rico para edição da linguagem;
- Possibilidade de definição do comportamento semântico para a linguagem, correlacionando interpretação e geração de código.

Nos dias atuais, alguns fabricantes já se consolidaram nesta área, como é o caso da MetaCase, com o MetaEdit++, Fundação Eclipse com o EMF/GMF e Microsoft através DSL Tools. (ROSA, AMARAL e BARROCA, 2009)

2.1.3.1 Microsoft DSL Tools

Esta ferramenta é incorporada a conceituada IDE Visual Studio, atualmente apresentada na versão 2010. Com essa ferramenta, é possível realizar a definição de uma DSL através de uma interface gráfica, apresentada na figura 3, e tem como conceitos-chave classes e relacionamento de domínio.

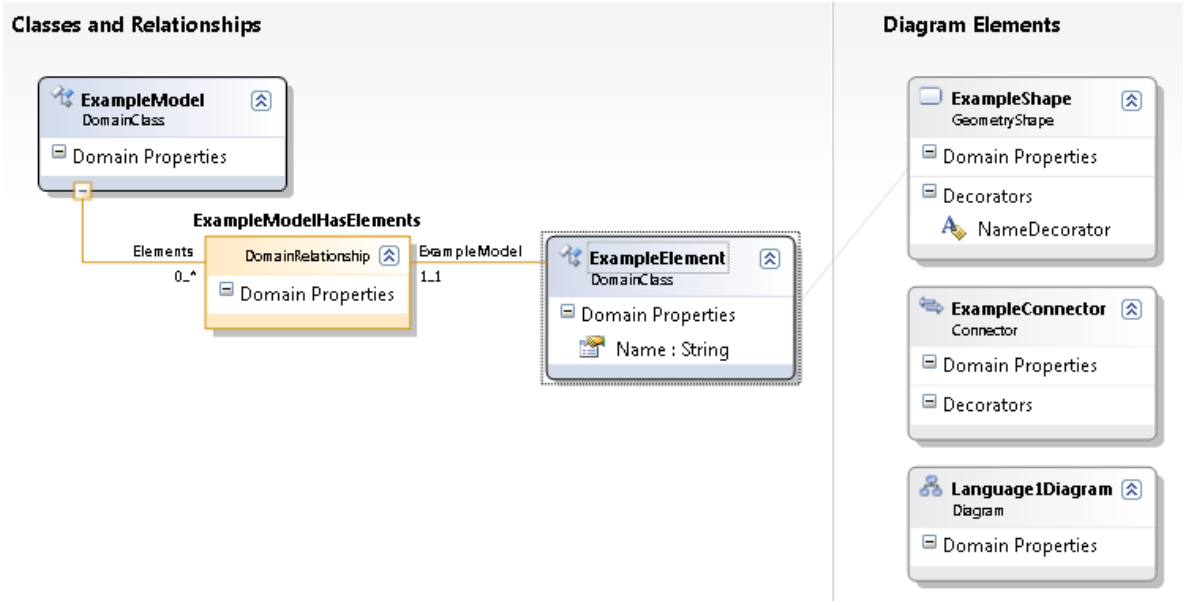

Figura 3 – Interface para criação de DSL a partir do Microsoft DSL Tools

Toda interpretação da DSL e geração de código é realizada através do mecanismo incorporado ao Visual Studio, denominado *Text Template Transformation Toolkit* ou simplesmente T4.

A própria empresa, Microsoft, utiliza essa ferramenta para a geração de DSL, como é o caso da *Service Factory*, que foi criada para o contexto de aplicações orientadas a serviço. O público-alvo desta DSL são os desenvolvedores ou arquitetos de software, pois tem como objetivo principal a criação de uma arquitetura orientada a serviços e definição dos seus serviços.

Esta linguagem tem como elementos-chave para a descrição de uma arquitetura orientada a serviços os seguintes itens:

- *Service*: Elemento que define um serviço.
- *ServiceContract*: Elemento que define a camada de interface do serviço.
- *Operation*: Define uma operação do serviço.
- *Message*: Define uma estrutura de dados que será utilizado como parâmetros ou dados de retorno de uma operação.

Na figura 4 pode-se ver todos estes elementos representados para uma arquitetura que descreve o gerenciamento de materiais. Nela é apresentado o serviço “MaterialMgmt” que é descrito, através do contrato “MaterialMgmtContract”, com duas operações “GetPartDemand” e “SetPartDemand”, essas possuem

mensagem que define os dados de entrada e saída como a mensagem “GetDemandRequest” que representa os dados da requisição da operação “GetPartDemand”.

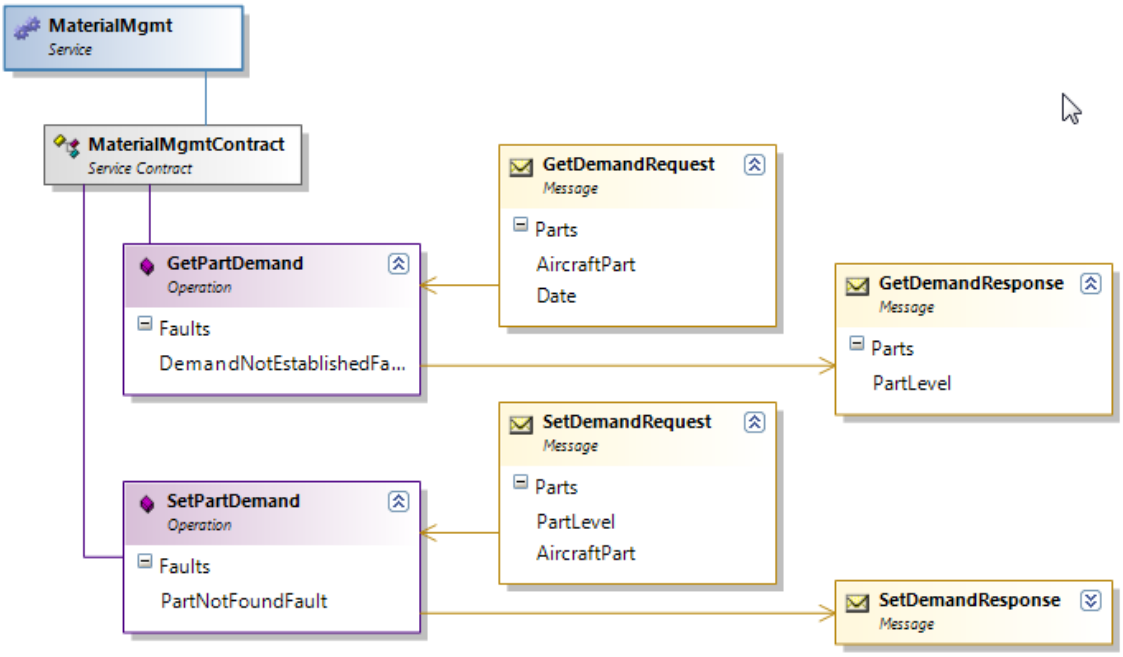

Figura 4 - Descrição do serviço “MaterialMgmt” utilizando a linguagem Microsoft Service Factory

Outra DSL criada pela Microsoft é o *Windows Workflow Foundation*, que apresenta uma interface gráfica para definição de fluxos de trabalho e também é voltada para aplicações orientada a serviço. Essa DSL é voltada para o desenvolvedor, pois apresenta termos tecnológicos que fogem do domínio dos profissionais de negócio, porém com o uso desta DSL é possível criar novas DSL que permitam que os profissionais de negócio sejam capaz de realizar a definição de um processo de negócio utilizando fluxos de trabalho.

2.2 Arquitetura Orientada a Serviços

Para o entendimento deste trabalho é necessário a clara definição do significado de SOA, porém devido a grande quantidade de definições propostas e grandes diferenças entre as mesmas, a busca de seu significado é um trabalho árduo (FUGITA, 2009). Para tanto, este capítulo apresenta a definição baseando-se no modelo de referência OASIS (2006), Papazoglou (2003) e Thomas Erl (2005).

2.2.1 Definição

Arquitetura Orientada a Serviços (SOA) pode ser definida, de maneira geral, como um paradigma para organização e uso de competências distribuídas que estão sob o controle de diferentes domínios proprietários (OASIS OPEN, 2006) e faz parte de um paradigma denominado Computação Orientada a Serviços (SOC) que tem os serviços como elemento fundamental para o desenvolvimento de aplicações (PAPAZOGLOU, 2003).

Segundo OASIS (2006), estes serviços são o mecanismo de acesso a um conjunto de uma ou mais competências que são definidas como um caminho para solucionar problemas enfrentados no âmbito de negócios. Essas competências são oferecidas por entidades denominadas Provedores de Serviço às entidades que possuem a necessidade, denominadas Consumidores de Serviço. Para que esse processo ocorra foram definidos 3 conceitos bases:

- Visibilidade: definida como a capacidade de Consumidores e Provedores se comunicarem entre si. Através das descrições do serviço que possui semântica e sintaxe compreensível.
- Interação: é o processo de execução de uma competência, através de troca de mensagens, ligadas a um contexto de execução.
- Efeitos: toda interação é realizada objetivando um efeito no processo de negócio, este podendo ser uma simples resposta a uma requisição ou a mudança de um estado compartilhado entre os participantes do contexto de execução.

O acesso a um serviço é realizado através de uma interface que permite a disponibilização de informações necessárias para a interação com um consumidor,

devido a esta característica, que oculta a forma de implementação do serviço exibindo apenas dados necessários, os serviços são definidos como caixa preta.

Além disso, Papazoglou (2003) define outro participante no processo de interação dos serviços, esse é denominado Registro de Serviço que é onde os Provedores publicam as descrições dos serviços que serão acessados pelos Consumidores, a figura 5 apresenta este relacionamento entre os participantes.

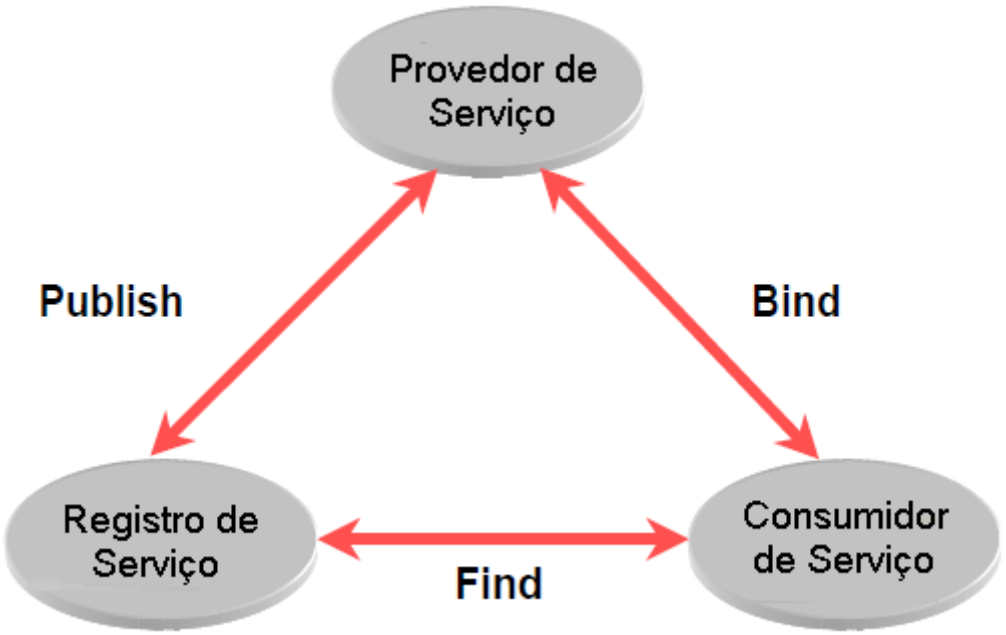

Figura 5 - Processo de interação dos serviços. (Adaptado de PAPAZOGLOU, 2003)

Quando a estratégia de implementação SOA é realizada através de Web Services (WS) normalmente o Registro de Serviço é implementado pelo padrão *Universal Description, Discovery and Integration* (UDDI).

Apesar de SOA não se restringir a utilização de Web Services, atualmente sua implementação usa essa tecnologia como base, juntamente como a especificação WS-* (WS-star), devido ao fato de que com esses é possível atingir algumas características de uma arquitetura de serviços, definidas por Thomas Erl (2005):

- Aumento na qualidade do serviço
- Baseado em padrões abertos
- Promove descoberta (*WS-Discovery*)

- Promove federação (*WS-Federation*)
- Reuso
- Extensibilidade
- Implementação de uma camada de abstração
- Promove baixo acoplamento para empresas

Com isso, Thomas Erl (2005) define SOA como uma forma de arquitetura tecnológica que adere aos princípios de orientação a serviço e que possui potencial para suportar e promover estes princípios para os processos de negócio e a automação de domínios de uma empresa quando realizada através de uma plataforma tecnológica como Web Services.

Analisando as definições apresentadas na literatura é possível perceber que apesar de conceitualmente apresentarem-se de maneira semelhante, eles definem SOA em contextos diferentes, tal como Thomas Erl (2005) que apresenta os benefícios e características de SOA usando Web Services.

2.2.2 Princípios da orientação a serviços

A orientação a serviços usa a teoria que é baseada na ideia da divisão de um problema em uma série de assuntos específicos, conhecida como Separação de Assuntos (ERL, 2005). Além da orientação a serviços muitos outros paradigmas fazem uso do mesmo conceito como é o caso da programação baseada em componentes que utilizando componente para obter esta separação (FUGITA, 2009).

No caso da orientação a serviços isto não é diferente, pois seus princípios foram idealizados com o objetivo de atingir a ideia de separação de assuntos (ERL, 2005). Esses princípios são descritos a seguir.

Reuso

SOA tem a capacidade de reuso de serviços intrínseca em suas características, mesmo que isto não seja um requisito do projeto. (ERL, 2005)

Este princípio facilita todas as formas de reuso desde a interoperabilidade entre empresas até a criação de serviços utilitários. No contexto de Web Services é possível implementar indiretamente este princípio através do uso de *SOAP Headers*. (ERL, 2005)

Contrato

Para usar um serviço é necessário que informações relacionadas a ele estejam disponíveis para o Consumidor dos Serviços, para tanto a orientação a serviços define o principio de contrato que define uma semântica para o serviço apresentando uma definição formal de (ERL, 2005):

- Protocolo de comunicação.
- Operações e seus dados de entrada e resposta.
- Regras e características de um serviço ou operações

Com o uso de padrões XML, necessários para a implementação através de Web Services, esta definição é realizada através de documentos como, WSDL e *Schemas* XSD.

É importante que contratos apresentem apenas os dados necessários para que o Consumidor de Serviços tenha a capacidade de entender e consumir o serviço.

Baixo acoplamento

Baixo acoplamento é uma condição em que um serviço adquire o conhecimento de outro se mantendo independente desse. Esta condição é atingida através do uso do princípio de contratos. (ERL, 2005)

Abstração

Este princípio refere-se à capacidade de um serviço omitir detalhes de sua implementação permitindo que atuem como uma caixa preta.

A granularidade das operações de um serviço está diretamente relacionada ao alcance e natureza das funções que estão sendo expostas por um serviço. (ERL, 2005)

Composição

Serviços devem possuir facilidade de composição, isto é, devem ser projetados de forma que possam participar de composições com outros, formando serviços compostos, ou ser orquestrados no contexto de um processo de negócio. Para isso, devem ser desacoplados e suas operações devem ser as mais atômicas possíveis. (FUGITA, 2009)

Autonomia

O princípio da Autonomia permite que um serviço tenha o controle total sobre sua própria lógica e tenha a capacidade de exercer auto governança.

. A autonomia pode-se dar em dois níveis diferentes: (ERL, 2005)

- Autonomia no nível de serviço: os limites entre os serviços são distintos, mas eles ainda podem compartilhar recursos encapsulados, como bases de dados e sistemas legados.
- Autonomia pura: a lógica e os dados encapsulados estão sob o controle do serviço, não sendo compartilhada com outros.

Independência de estado

Serviços devem minimizar a quantidade de informações de estado que são gerenciadas assim como o tempo que estas informações ficam sobre o gerenciamento. Caso isto não ocorra, é provável que a capacidade de estar disponível para outros Consumidores do serviço esteja comprometida.

A capacidade de ser independente de estado é uma condição importante para serviços e esta promove o reuso e escalabilidade.

Localização

Os serviços devem ser localizáveis, o que significa que suas interfaces devem ser, de alguma forma, publicadas e tornadas disponíveis para os potenciais Consumidores de Serviço. Na arquitetura SOA básica, essas interfaces são

publicadas em um Registro de Serviços, de onde elas podem ser buscadas pelos Consumidores. (FUGITA, 2009)

2.2.3 Processo de desenvolvimento SOA

Antes de descrever como é possível a inclusão do uso de DSL no processo de desenvolvimento de uma aplicação SOA é necessário entender como é realizado o desenvolvimento da mesma.

O ciclo de vida de um projeto SOA inicia-se na modelagem do negócio passando pelas fases de Análise e Projeto até as atividades de construção, conforme descrito na figura 6. (FUGITA, 2009)

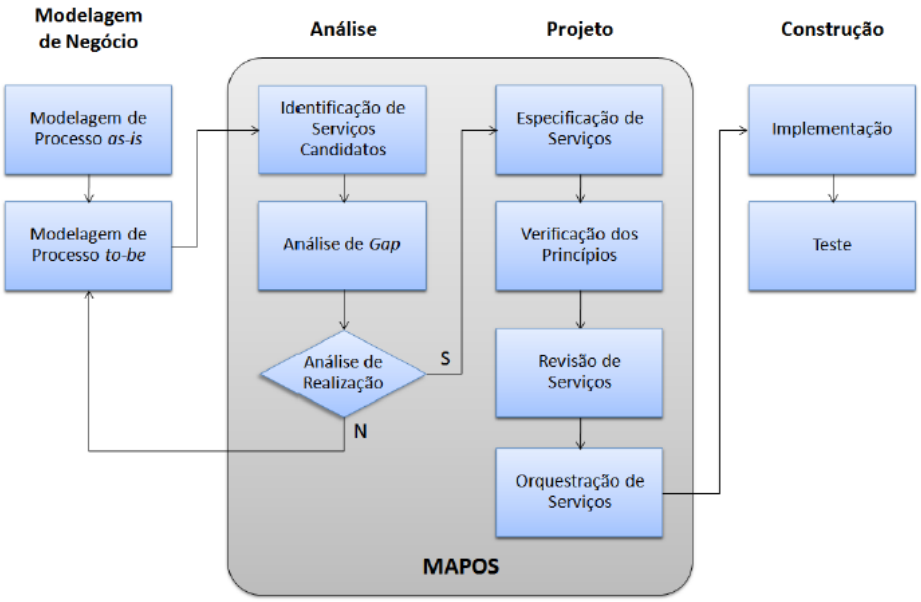

Figura 6 - (MAPOS) Método de Análise e Projeto Orientado a Serviços (FUGITA, 2009).

A fase de modelagem compreende o entendimento sobre as necessidades de negócio e levantamento de requisitos e possui os seguintes artefatos gerados: (FUGITA, 2009).

- Modelo de processo as-is: demonstrando como estão os processos atualmente.
- Modelo de processo *to-be*: Modelo com as alterações propostas, ao modelo *as-is*, com o intuito de promover melhorias de desempenho e reduzir custos.

Na fase de Análise, são identificados serviços candidatos a partir dos modelos de negócio fornecidos. Ainda nesta fase, decide-se como cada serviço será realizado, analisando-se possibilidades de reuso para cada um, chegando a um conjunto de serviços finais. (FUGITA, 2009)

Após a análise e definição dos serviços inicia-se a fase de Projeto que tem como objetivo a criação das especificações de serviços que sejam aderentes aos requisitos de negócio. Essa fase tem como artefatos resultantes a orquestração de serviços e as especificações técnicas que servirão para a fase de Construção.

Na fase de Construção, são desempenhadas atividades com o intuito de se obter os serviços executáveis devidamente integrados e testados, prontos para serem implantados. (FUGITA, 2009)

2.2.4 Relação com o paradigma orientado a objetos

O uso do paradigma orientado a serviços não significa alterar a maneira como é realizado o desenvolvimento de software, ignorando todo o conhecimento obtido através do aprimoramento de outros paradigmas, pois orientação a serviços é uma evolução do paradigma orientado a objetos junto ao desenvolvimento baseado em componente. (FUGITA, 2009)

A orientação a objetos normalmente é usada para a implementação da lógica encapsulada em um serviço e princípios da orientação a serviços foram formulados baseando-se no conceito da mesma. (FUGITA, 2009)

Apesar disso, conceitos como baixo acoplamento apresentam características diferentes em cada um dos paradigmas. Na orientação a serviços esse conceito é definido com a capacidade de um serviço ser consumido independente de onde seu Provedor de serviço esta localizado, já na orientação a objetos é descrito como a capacidade de minimizar as dependências e maximizar o reuso, porém por definição do paradigma as classes possuem relacionamento entre si, como Herança e Agregação (FUGITA, 2009).

2.3 Tecnologias XML

Originalmente desenvolvida com o objetivo de suprir a necessidade na publicação eletrônica em larga escala, a XML vem se tornando um padrão para troca de dados variados na Web (W3C, 2010a). Com sua popularização, novas tecnologias surgiram com o intuito de adicionar novos recursos. Neste capítulo serão apresentadas as tecnologias necessárias para a transformação e definição de documentos XML.

2.3.1 Esquema XML (XSD)

Esquemas XML tem o propósito de definir um vocabulário que especifique o conteúdo de documentos XML e que seja facilmente lido por humanos e computadores.

Com este vocabulário é possível realizar a validação do documento contra o esquema. Essa validação é um importante item para o processo de garantia da qualidade do software (W3C, 2009). Documentos XML gerados por este esquema são denominados Instância.

O vocabulário define dois tipos de dados, complexos e simples. Basicamente a diferença entre estes é a possibilidade de possuir elementos e atributos vinculados ao seu conteúdo que pode ocorrer somente em tipos complexos. A figura 7 apresenta um exemplo da definição de cada um dos tipos de dados disponíveis, sendo que a primeira apresenta o elemento complexo “Endereço” definido com quatro elementos internos que utilizam tipos de dados simples.

```
<xsd:complexType name="Endereco">
  <xsd:sequence>
    <xsd:element name="logradouro" type="xsd:string"/>
    <xsd:element name="cidade"   type="xsd:string"/>
    <xsd:element name="estado"  type="xsd:string"/>
    <xsd:element name="cep"    type="xsd:decimal"/>
  </xsd:sequence>
</xsd:complexType>
```

Figura 7 - Exemplo de utilização da definição através de tipo de dados complexo e simples (W3C, 2009)

Além disso, o esquema define o número de ocorrências que um elemento pode ocorrer em um documento, sequências que aparecem, restrições no formato do dado, obrigatoriedade, enumeradores e *namespace.*

2.3.2 Transformação

Em projetos que usam tecnologias XML é comum a necessidade de transformação do conteúdo de uma fonte de dados para estruturas diferentes. Para realização dessas transformações, são usadas duas tecnologias, XSLT e XPath definidas pela W3C, que é um consórcio internacional no qual organizações filiadas, uma equipe em tempo integral e o público trabalham juntos para desenvolver padrões para a Web. (W3C, 2008)

A W3C define o processo de execução das transformações de documentos usando estas tecnologias, apresentado na figura 8. Esse processo, basicamente obtém informações de fonte de dados externas que são extraídas através do *XPath 2.0 Data Model Builder* gerando um modelo interno que é executado através de um processador XSLT 2.0 transformando as informações em um documento XML.

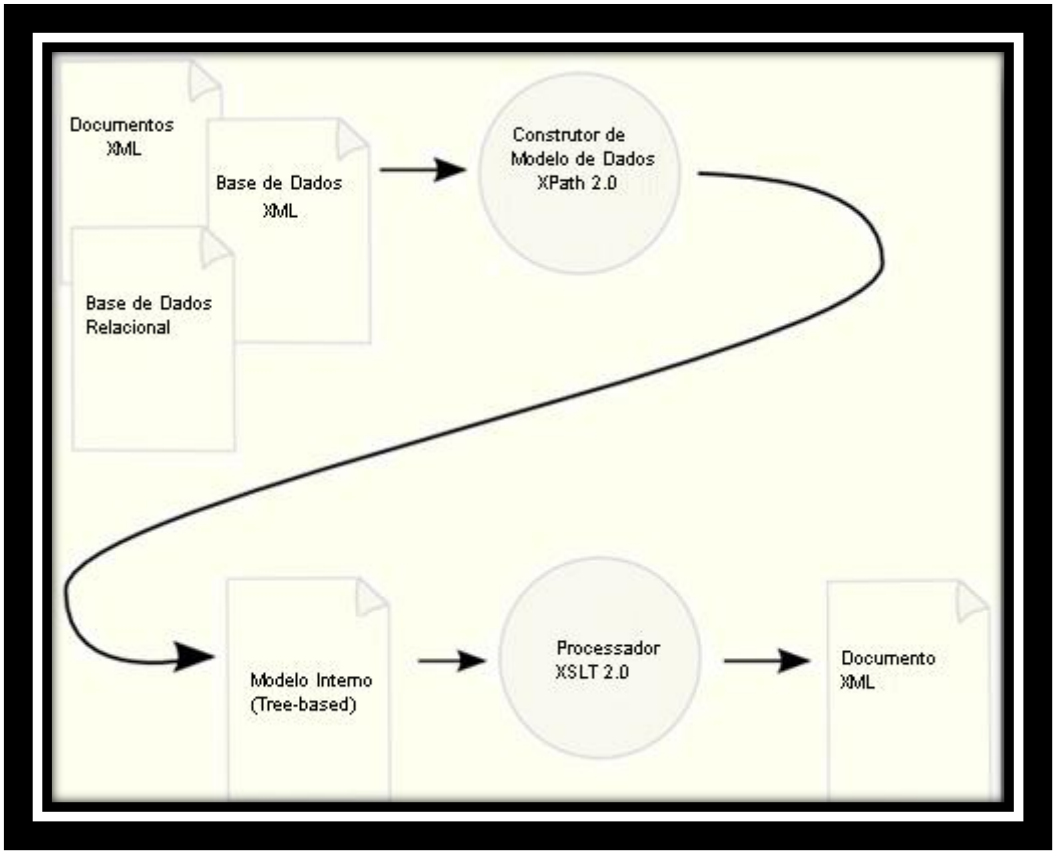

Figura 8 - Processo de execução de transformação. (Adaptado de W3C, 2010b)

2.3.2.1 XSLT

O XSLT (*XML StyleSheet Language Transformations*) é uma linguagem para transformação de documentos XML em outros documentos, podendo estes serem XML, HTML ou simplesmente textos. Essa linguagem é expressa através de documentos XML bem-formados e suas transformações são chamadas *stylesheet*.

A execução dessas transformações é realizada através da instanciação de regras de *template* que possuem duas partes. A primeira define em qual nó do documento de origem este *template* está atrelado e a segunda define as regras para geração do documento transformado. Esta estrutura permite que essas regras sejam reutilizadas por documentos com estruturas similares. (W3C, 1999)

As regras de geração podem ser descritas através de elementos literais do documento resultante ou através de instruções definidas pelo XSLT *namespace*, **http://www.w3.org/1999/XSL/Transform**, apresentadas na tabela 1.

Tabela 1 - Lista de instruções do XSLT *Namespace*. (W3C, 1999)

Função	Descrição
xsl:apply-imports	Usado para invocar uma regra de *template* sobescrita.
xsl:apply-templates	Processa todos os nós-filho, incluindo nós de texto.
xsl:atribute	Usado para adicionar atributos para elementos.
xsl:attribute-set	Usado para criar um agrupamento nomeado de atributos.
xsl:call-template	Invoca um *template* através do nome.
xsl:choose	Inicia uma condição de numerosas alternativas, utilizado juntamente com xsl:when e xsl:otherwise.
xsl:comment	Cria um comentário no documento resultante.
xsl:copy	Realiza a cópia do nó atual.
xsl:copy-of	Insere um elemento do documento resultante sem converter em texto.
xsl:decimal-format	Declara um formato para números decimais para ser utilizado pela função xsl:format-number.
xsl:element	Usado para criar elementos no documento resultante.
xsl:for-each	Usado para percorrer o documento de origem baseado em condições.
xsl:if	Usado para criar uma condição.
xsl:import	Importa outro XSLT *stylesheet*.
xsl:include	Importa outro XSLT *stylesheet*.
xsl:key	Declara uma chave para se utilizada no documento inteiro.
xsl:message	Define mensagens para ser utilizadas na geração de textos.
xsl:namespace-alias	Declara um *alias* para o *namespace* vindo de outro *namespace*.
xsl:number	Insere um número formatado.
xsl:otherwise	Cláusula que indica que nenhuma condição foi contemplada.
xsl:output	Define como será realizado o processamento da mensagem de saída.
xsl:param	Usado para definir parâmetros para um função, como o xsl:call-template.
xsl:preserve-space	Usada para realizar a preservação de espaços em branco.
xsl:processing-instruction	Cria um nó de instrução de processamento.
xsl:sort	Realiza a ordenação através das chaves declaradas, xsl:key.
xsl:strip-space	Usada para realizar a preservação de espaços em branco.
xsl:template	Define um *template*.
xsl:text	Define o texto de um elemento.
xsl:value-of	Usado para extrair texto de variáveis ou nós do documento de origem.
xsl:variable	Define uma variável.
xsl:when	Define uma condição para o xsl:choose.
xsl:with-param	Define a utilização de um parâmetro para o xsl:call-template.

2.3.2.2 XPath

Como descrito anteriormente, o XSLT tem a capacidade de representar como o dado extraído será transformado, porém para que a transformação de um determinado elemento ocorra com sucesso, é necessário a extração dos dados do mesmo, e para isso, é necessário usar XPath em conjunto com o XSLT. O XPath tem como propósito a navegação e seleção de nós de um documento XML, através de uma notação de navegação de pastas.

Assim como na navegação de pasta, o resultado da execução das expressões XPath está vinculado ao contexto, isto significa que a expressão é executada em um nó específico, porém a linguagem oferece algumas expressões que permitem mudar o contexto da execução (MICROSOFT, 2010):

- "Contexto atual: expressões iniciadas com"./".
- Nó principal do documento: expressões pré-fixadas com "/".
- Nó principal do elemento: expressões iniciadas com "/*".
- Pesquisa recursiva: expressões pré-fixadas com "//".

Além disso, a linguagem oferece uma sintaxe para a filtragem dos dados selecionados. A figura 9 apresenta uma expressão que seleciona todas as compras realizadas onde o elemento "parcel" seja igual a "28-451". A linguagem permite também que atributos sejam utilizados nos filtros.

```
/transactions/purchase[parcel="28-451"]
```

Figura 9 - Exemplo de XPath

A linguagem oferece também operações que podem ser utilizadas em expressões, tais como as apresentas na tabela 2.

Tabela 2 - Exemplo de funções XPath. (Adaptado de W3C, 2010c)

Função	Descrição
Count	Realiza a contagem de nós definidos em uma expressão.
Concat	Concatena dois textos.
Compare	Compara dois textos e retorna -1, 0, 1 dependendo se o primeiro argumento é menor, maior ou igual ao segundo.

A lista completa de funções está disponível em: http://www.w3.org/TR/xquery-operators/.

2.4 Considerações do capítulo

Este capítulo apresentou o conceito de SOA e DSL e algumas tecnologias XML, estas informações serão necessárias para o entendimento de como é possível realizarmos a utilização do conceito de DSL em aplicações orientadas a serviço.

A apresentação do conceito SOA foi realizada, a todo o momento, utilizando Serviços Web como forma de implementação isto ocorreu devido ao fato de que a aplicação apresentada neste trabalho e implementações atuais seguem este paradigma.

Neste capítulo foi dado foco na *Microsoft DSL Tools* e *Windows Workflow Foundation* devido ao fato de que o experimento será realizado utilizando a linguagem C# como linguagem de execução, que é a linguagem base dessas DSL, porém é importante frisar que muitos outros Workbench de Linguagem estão disponíveis no mercado e estas não apresentam diferenças em relação à usabilidade.

3. LINGUAGENS DE DOMÍNIO ESPECÍFICO ORIENTADAS A SERVIÇOS

Nos últimos anos, a área de desenvolvimento de software vem usando em larga escala a linguagem de modelagem UML para descrever e especificar um sistema de software. Apesar disso, pesquisas apresentam que em projetos baseados no conceito SOA é necessário que a UML seja estendida para incorporar informações deste tipo de projeto. Esta extensão é realizada através da utilização de *Profile*, que permitem que elementos com nova semântica sejam incorporados ao modelo, através do uso de um meta-modelo.

Contudo, pesquisas recentes mostram que apesar da especificação através da UML ser uma prática consolidada, o uso de DSL se torna uma prática de menor custo, tempo e complexidade. Estas variáveis garantem que um produto esteja disponível ao mercado com mais rapidez (*Time to Market)*, que no atual momento do mercado é um dos grandes diferenciais que garantem o sucesso de um produto. (LINDBERG e THORIN, 2007)

3.1 Fluxos de trabalho

A proposta de processo de criação de DSL apresentada no próximo capítulo usa o conceito de fluxo de trabalho como base para a definição da DSL, que é definido como um modelo que representa e descreve a ordem de execução e relações de dependências entre atividades. As atividades são os elementos essências de um fluxo, pois são a representação das ações, humanas ou computacionais, realizadas em um processo de negócio empresarial.

Como já visto nos capítulos anteriores um serviço de uma arquitetura orientada a serviços é a representação de uma ou um conjunto dessas ações e isto nos leva a definição de que o uso de um fluxo de trabalho para descrição de um serviço é uma prática aceitável.

Além disso, as atividades necessárias para a criação de um fluxo de trabalho são de domínio de profissionais que detêm grande conhecimento de negócio e conhecimento técnico razoável.

Apesar dos possíveis benefícios do simples uso de um fluxo de trabalho para descrever um processo de negócio, é necessário que esses fluxos sejam executáveis através de uma infraestrutura computacional, ou seja, devem ser a representação do código executável de uma aplicação. Para suprir esta necessidade é necessário um ambiente executável do fluxo e para isto alguns fabricantes fornecem ferramentas.

3.2 Trabalhos relacionados

Este capítulo apresenta três trabalhos que realizam ou propõem um processo de criação de uma DSL. O primeiro realiza a comparação entre duas técnicas de modelagem – UML Profile e DSL,

O segundo trabalho define um processo de criação de DSL e a sua aplicação em três experimentos distintos. Neste trabalho também é apresentado pelos autores a estratégia de desenvolvimento de software dirigida por modelos, MDSD (*Model-driven software development*).

Já o ultimo trabalho apresenta a comparação entre uma linguagem baseada em UML com uma DSL. Estas linguagens são usadas por um departamento da ONU.

3.2.1 SOLA

Lindeberg e Thorin da Universidade de Göteborg, Suécia, apresentaram uma análise sobre as vantagens e desvantagem da criação de DSL para o contexto de SOA. (LINDBERG e THORIN, 2007)

Os autores comparam duas técnicas para o desenvolvimento desta solução:

- UML Profile usando *IBM Rational Software Modeler*.
- DSL usando Microsoft DSL Tools.

No processo de uso da DSL, inicialmente foi criada uma linguagem com o objetivo de modelar uma arquitetura orientada a serviços denominada *Service-oriented Language for Architectures* (SOLA). Nessa linguagem foram implementados os seguintes elementos de uma arquitetura, apresentados na figura 10:

- *Service* (Serviço)

- *ServiceInterface* (Interface de Serviço)
- *Message* (Mensagem)
- *ServiceClient* (Consumidor)
- *ServiceRepository* (Repositório)
- *ServiceAggregator* (Composição)
- *ServiceProvider* (Fornecedor)
- *Operation* (Operação do serviço).

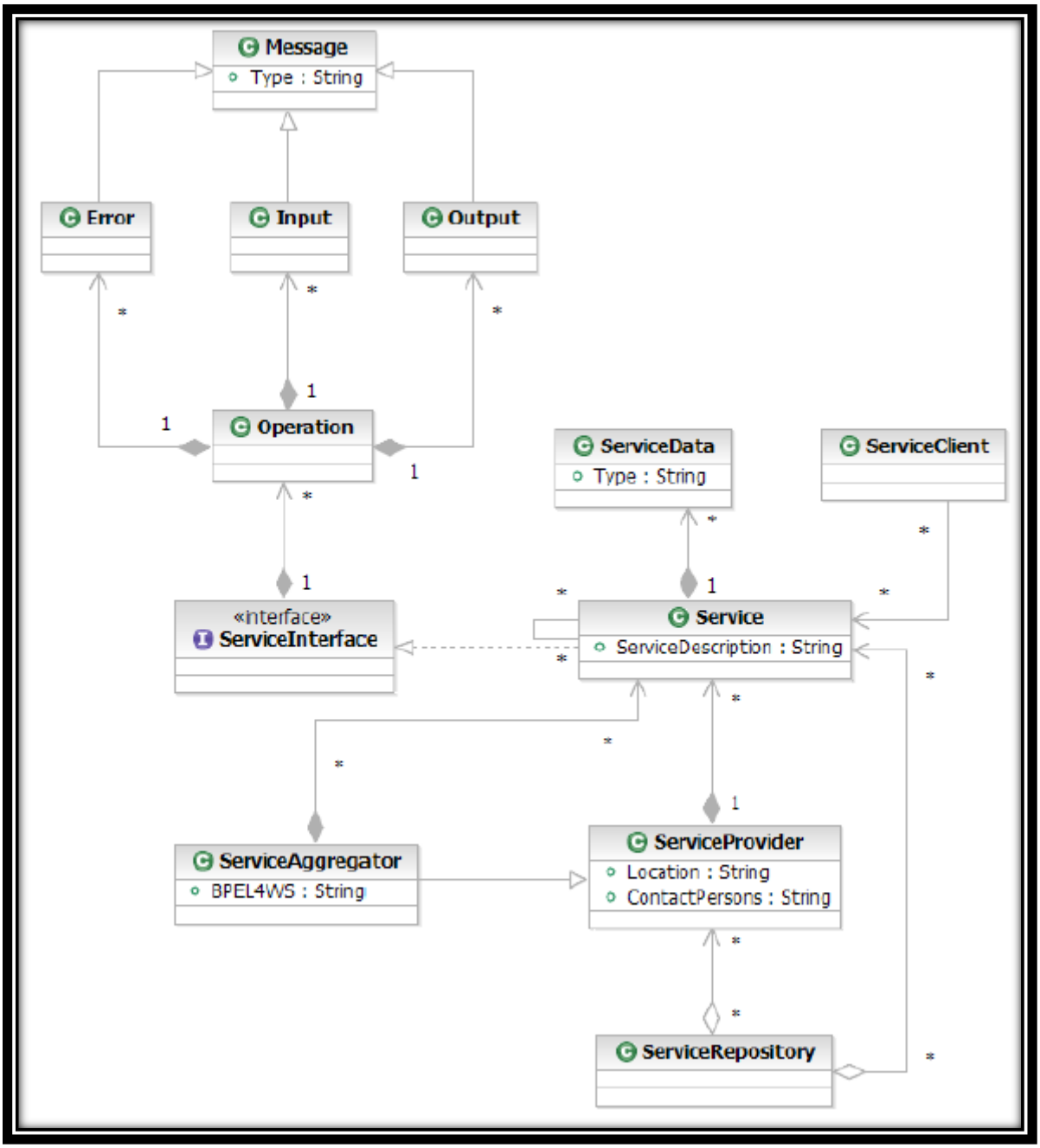

Figura 10 – Meta-modelo SOLA (LINDBERG e THORIN, 2007)

Os dados para a comparação das duas técnicas foram obtidos através de experimento que realizava a modelagem de uma aplicação SOA.

Neste experimento foi realizada a modelagem de uma aplicação SOA de exemplo em ambas as técnicas e realizada uma análise estática dessas.

O projeto de modelagem foi separado em três incrementos sendo que estes foram organizados de uma maneira que evitasse que o estudo ocorra somente em uma das técnicas. O primeiro incremento usa primeiramente SOLA e depois a *UML Profile*, já no segundo a ordem é invertida e no terceiro a SOLA é novamente usada primeiro conforme apresentado na Figura 11.

Figura 11 - Processo incremental do Experimento. (Adaptado de LINDBERG e THORIN, 2007)

A organização dos dados obtidos foi realizada através da técnica de *Goal Question Metric* (GQM), que sugere que perguntas sejam separadas em métricas mensuráveis que juntas chegam ao resultado de uma avaliação. Os autores definiram que para realizar a avaliação das duas técnicas seriam necessárias 3 métricas.

- Homem/hora por tarefa
- Número de elementos disponíveis
- Número de elementos disponíveis em uma única tela no processo de modelagem.

Com isto, os autores chegaram à conclusão que o esforço de criação de uma DSL para SOA, através da Microsoft DSL Tools, é aproximadamente 28% menor do na técnica que utiliza UML Profile. Além disso, foi identificado que a utilização da SOLA garantiu que o esforço para a modelagem de uma aplicação fosse aproximadamente 35% menor. Apesar disso, os autores concluíram que o melhor cenário é a aquele no qual a empresa encontra um balanço entre o uso de

linguagens de propósito geral e linguagens específicas, pois uma linguagem muito geral, como o caso da UML Profile para SOA, sofre com a complexidade no processo de modificação e a criação de muitas linguagens específicas pode causar um aumento no esforço necessário para o desenvolvimento da solução.

Contudo, os autores deixaram claro que o propósito do trabalho é avaliar a maturidade da ferramenta Microsoft DSL Tools e não a sua incorporação ao processo da empresa.

3.2.2 DSL baseada em MDSD

Oberortner, Zdun e Dustdar apresentaram em seu trabalho (OBERORTNER, ZDUN e DUSTDAR, 2008) um estudo exploratório sobre o uso de DSL para expressar modelos que permitem a geração de serviços que realizem tarefas individuais de um processo.

Os autores realizaram o desenvolvimento das DSL apresentadas nesse estudo através da estratégia de desenvolvimento de software dirigida por modelos, MDSD (*Model-driven software development*). Esta estratégia define o uso de modelos como o artefato principal no desenvolvimento, isto significa que a geração aplicações completas é realizada através das transformações desses modelos em código executável.

Com isso, os autores apresentaram a estratégia de desenvolvimento desta DSL, primeiramente ocorreu à separação de assuntos em dois níveis, com o objetivo de garantir um maior entendimento de cada *stakeholders*.

- Alto-nível: para especialista de domínio, que não necessitam de informações técnicas.
- Baixo-nível: uma extensão do alto-nível específica para especialistas técnicos.

Além dessa separação, os autores definiram que cada nível de DSL representam modelos de linguagens apropriados, esses modelos podem ter múltiplas instancias e múltiplas sintaxes DSL, conforme aprestando na figura 12.

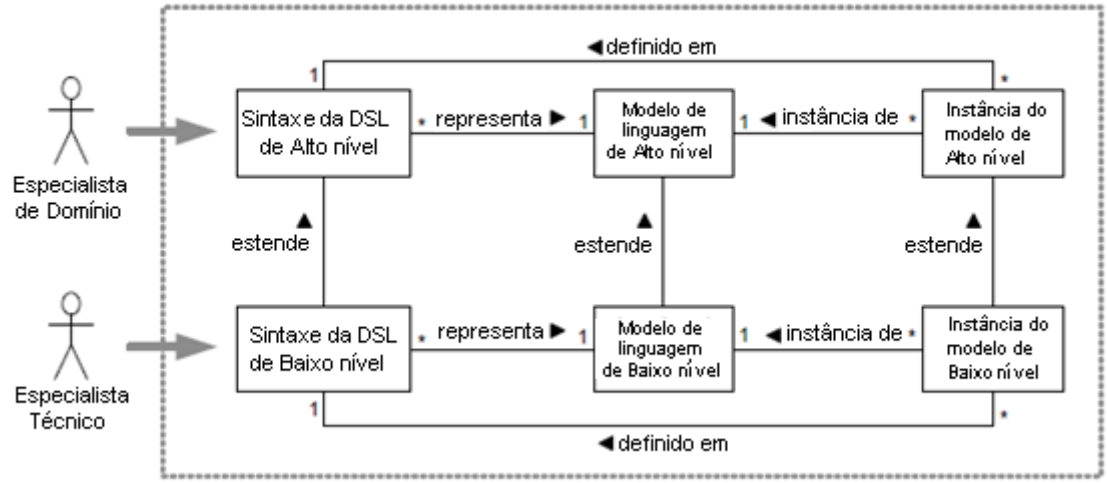

Figura 12 - MDSD – DSL. (Adaptado de OBERORTNER, ZDUN e DUSTDAR, 2008)

Para obter os resultados desta estratégia foram realizados 3 experimentos, o primeiro os conceitos básico de uma SOA dirigida por processos, que os autores definem como uma arquitetura onde os serviços realizam tarefas ou passos de um processo individualmente e que lidam com inúmeros conceitos tais como:

- Orquestração de processos de negócio
- Informação em processos
- Colaboração entre processos e serviços

O segundo realiza adição, na DSL criada no primeiro experimento, dos conceitos de Transações de longa duração e Interação humana. Já o terceiro experimento tem como objetivo a criação de DSL para o controle do fluxo de páginas WEB, figura 13.

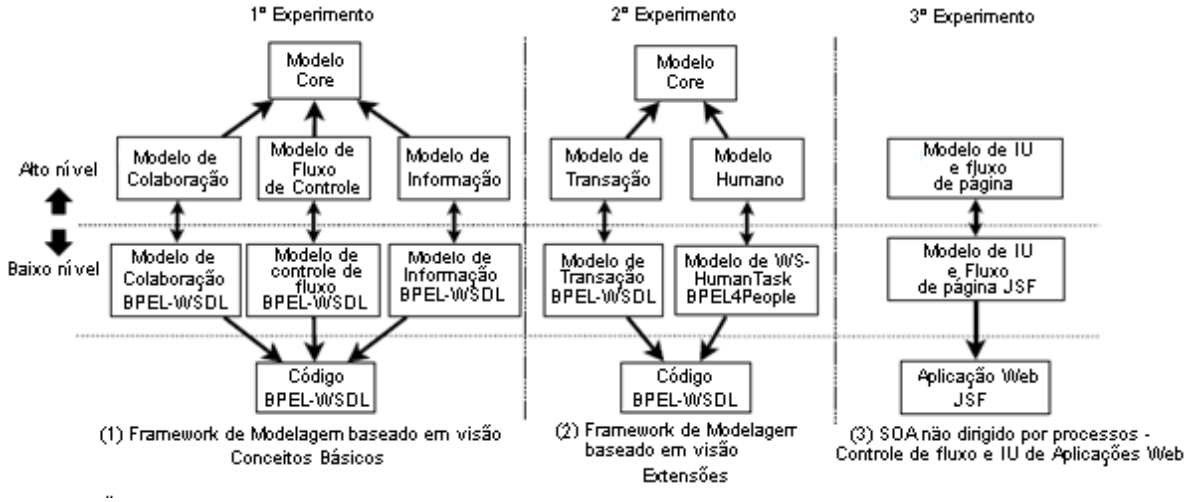

Figura 13 - Visão geral dos experimentos. (Adaptado de OBERORTNER, ZDUN e DUSTDAR, 2008)

Como resultados, os autores identificaram que o uso dessa estratégia foi válido nos três experimentos e que essa permitiu o suporte a diferentes interessados no projeto com diferentes conhecimentos. Outro aspecto importante é o fato que as DSL baseadas em MDSD melhoraram a capacidade de leitura e entendimento dos interessados além de reduzir a complexidade da SOA dirigida por processos.

Apesar destes aspectos, foram identificados alguns compromissos de projeto no processo de desenvolvimento usando DSL.

- Quanto mais detalhada a separação de um modelo de linguagem em outras, mais complexo o modelo de integração destas linguagens.
- Quanto mais complexo os pontos de integração, maior dificuldade os interessados terão em entender e ler a DSL ou seus modelos de linguagem.

3.2.3 Linguagem de coreografia de negócios

Motal, Zapletal e Werthner apresentaram a Linguagem de coreografia de negócios (BCL), uma DSL criada como alternativa a UML Profile desenvolvida pelo *United Nations Center for Trade Facilitation and Electronic Business* (UN/CEFACT) que tem como intuito a descrição de coreografias de processos B2B. Esta notação é denominada Metodologia de Modelagem da UN/CEFACT (UMM).

Segundo os autores, coreografia global descreve um processo de um ponto de vista neutro capturando os comportamentos observáveis entre dois ou mais parceiros e essas tem ganhado muito atenção no contexto de orientação a serviços.

Os motivos que levaram os autores a criação desta DSL foram:

- Devido ao fato de ter com base a UML, muitos improvisos foram necessários.
- Usuários-chave acreditavam que a UMM é muito complexa.
- A legibilidade e visualização gráfica dos modelos foram criticadas.
- Grande número de modelos resultantes.

A DSL criada tem como base os conceitos definidos presentes na UMM acrescentando conceitos do BPMN:

- Colaboração de negócio (C1): significa um processo de negócio que é conduzido por um ou mais participantes que governa o fluxo entre transações de negócios.
- Transação de negócio (C2): é o bloco que sincroniza o estado entre parceiros
- Padrões das transações de negócio (C3): define o tipo de interação entre aplicações parceiras, estas podendo ser:
    - Unidirecional: Notificação, distribuição de informações.
    - Bidirecional: Request/Response, Query/Response, Request/Confirm, Transações comerciais).
- Parâmetros de qualidade de serviço (C4): define restrições de segurança e tempo excedido.
- Documentos de negócio (C5): define informações trafegadas em transações de negócio.
- Estados compartilhados (C6): conceito necessário para que mais de um participante tenha a capacidade de conduzir o processo e muitas vezes esses estados definem o fluxo de controle em uma colaboração.
- Reuso de transações de negócio (C7): Reutilização de transações de negócio em diferentes colaborações de negócio.
- Mapeamento de papéis (C8): Permite a abstração dos parceiros de negócio.
- Eventos temporais (C9): Eventos que ocorrem após certa condição de tempo seja alcançada.
- Compensações (C10): Processo que permite desfazer uma transação de negócio caso alguma exceção ocorra.
- Event-based XORs (C11)

A Figura 14 apresenta uma colaboração de negócio (C1), *Order from Quote*, modelada pela DSL. Nesta colaboração são apresentadas 5 transações de negócio, ex. *RequestForQuote*, representadas por uma bloco arredondado. Nestes blocos são representados os participantes da transação e estes participantes são definidos no bloco *Participants*, no exemplo *buyer e seller*, e em cada transação assumem o papel de iniciar a transação ou responde-la.

Estas transações possuem documentos de requisição e de resposta, no caso da transação *PlaceOrder* estes documentos são *OrderEnvelope* (Requisição) e *OrderAcceptanceEnvelope* e *OrderRejectionEnvelope* (Resposta) e o padrão de processamento que esta irá ocorrer, Unidirecional e Bidirecional, que é representado por flechas.

Além disso, a figura apresenta o fluxo de processamento desta colaboração através de links entre as transações que são iniciadas após a transação atual obter resultado ou, como no caso do *RequestForQuote*, tiver o período máximo de execução expirado.

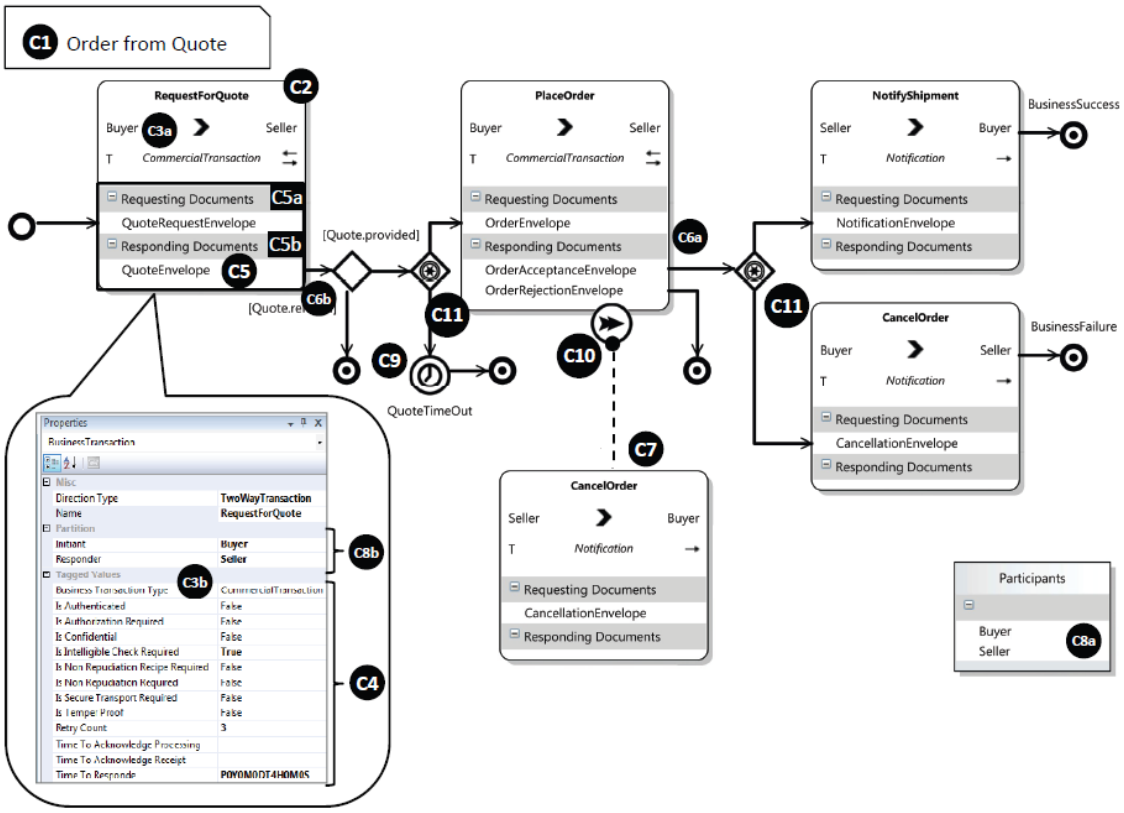

Figura 14 - Uso da BCL: Ordem de Compra com cotação (MOTAL, ZAPLETAL e WERTHNER, 2009)

Como resultado, os autores identificaram os seguintes benefícios da BCL comparando com a UMM:

- Maior facilidade de entendimento.
- Redução na quantidade de elementos de modelagem
- Redução na quantidade de modelos
- Validação do modelo simplificada

3.3 Considerações do capítulo

Como o objetivo deste trabalho é analisar as vantagens e desvantagens da utilização de DSL para SOA, os resultados apresentados pelos trabalhos relacionados são muito importantes para as considerações finais deste trabalho.

Como foi possível perceber pela leitura dos trabalhos relacionados, inúmeras abordagens para a criação de uma DSL estão disponíveis e todas apresentam vantagens e desvantagens.

Além disso, foi possível notar que com a utilização desta abordagem, fluxos de trabalho, é possível o uso de ferramentas disponíveis atualmente, o que torna o processo de adaptação muitas vezes menos problemático.

4. PROPOSTA DE PROCESSO DE CRIAÇÃO DA DSL

Neste capítulo é apresentada uma proposta de processo de criação de DSL, que utiliza o conceito de fluxos de trabalho, para o desenvolvimento de uma aplicação SOA.

4.1 Processo de criação de DSL

Seguindo o ciclo de vida de um projeto SOA descrito no capítulo 2, após a atividade de “Análise de Realização” é necessária a definição de como as atividades serão organizadas para que este serviço execute as atividades do processo de negócio, isto é denominado orquestração de serviço. É a partir deste momento que a proposta de processo de criação de uma DSL usando fluxos de trabalhos, apresentada neste capítulo, inicia-se.

Baseando-se na proposta de separação de assunto em níveis, apresentada em OBERORTNER, ZDUN e DUSTDAR (2008), o processo proposto separa os elementos da DSL em dois níveis.

- MACRO
- MICRO

A primeira atividade do processo proposto é a “Identificação das atividades do processo”, seguindo para a “Modelagem das Micro Atividades” e “Modelagem das Macro Atividades”, conforme apresentado na figura 15.

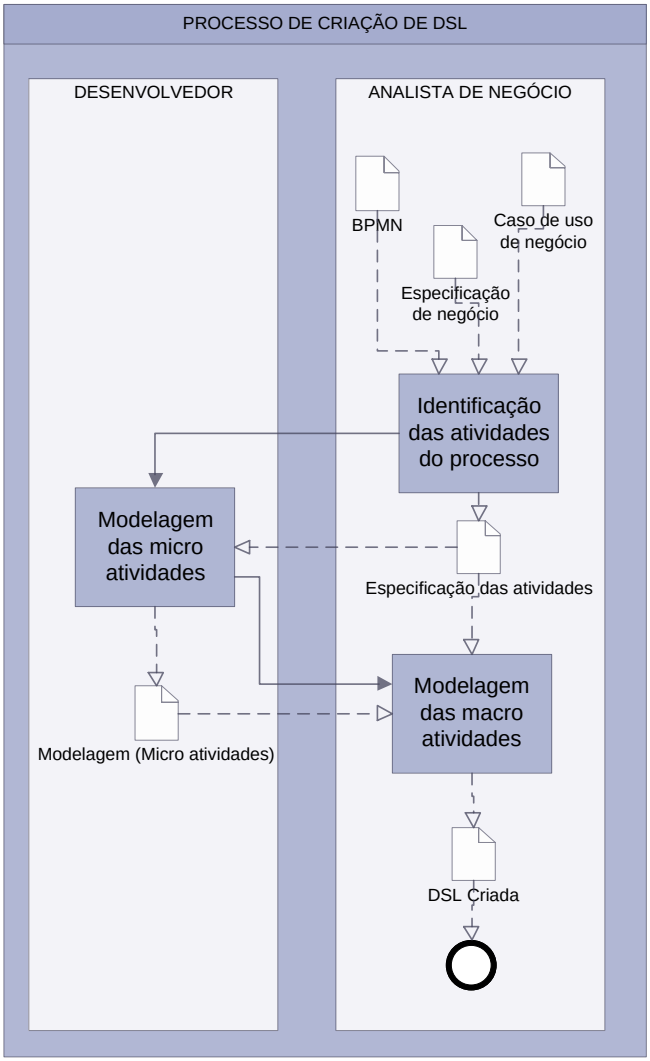

Figura 15 - Processo de criação de DSL

4.1.1 Atividades do processo

Com o intuito de facilitar o entendimento do processo todo o detalhamento das atividades é apresentado seguindo a estrutura descria abaixo:

- Descrição: detalhamento da atividade.
- Pessoas envolvidas: pessoas responsáveis pela realização da atividade e seu percentual de atuação.
- Artefatos de entrada: todos os artefatos externos que são utilizados na realização da atividade.
- Artefatos de saída: todos os artefatos gerados pela atividade.
- Ferramentas usadas: descreve as ferramentas que podem ser utilizadas para a realização de uma atividade.

4.1.1.1 Identificação das atividades do processo

Descrição:

O primeiro passo do processo de criação de uma DSL é criar a especificação e categorização das atividades do processo de um serviço. Com isso, esta atividade tem como objetivo identificar dois tipos de atividades:

- Macro atividades: agrupamento de micro atividades que representam uma atividade de negócio, por exemplo: Análise de informações do cliente.
- Micro atividades: atividades atômicas que compõem a execução de apenas uma ação, por exemplo: Enviar E-mail.

Para que seja possível a definição do processo de um serviço é preciso identificar quais serviços devem ser especificados. Para isto, é necessário que o BPMN usado como artefato de entrada represente estes serviços através de Services Tasks ou então apresente apenas o processo de um serviço específico.

Com o intuito garantir uma maior capacidade de reuso em desenvolvimentos futuros ou até mesmo manutenções e permitir a criação de uma DSL mais consistente, é recomendada a definição dos processos de todos os serviços identificados na solução permitindo assim.

Com as atividades devidamente categorizadas o analista com conhecimento de negócio deverá realizar a criação de um documento que represente o relacionamento entre estas atividades e as tarefas das atividades.

Esse documento será utilizado como documento base para o processo de verificação da DSL criada e devido a isto é necessário que o profissional que criará o documento esteja com suas atenções voltadas exclusivamente para esta atividade.

A especificação de uma atividade pode ser realizada através dos seguintes campos:

- Nome da atividade
- Tipo de atividade
- Atividades relacionadas
- Fluxo da atividade

O Nome da atividade define o identificador da atividade, enquanto o campo Tipo de atividade define a granularidade da atividade dentro do processo, macro ou micro, e esta informação é necessária para a execução das próximas atividades do processo. Já o campo Fluxo da Atividade descreve todas as tarefas da atividade e o campo Atividades relacionadas define todas as atividades que a mesma se relaciona. Um modelo de especificação com os campos descritos é apresentado na tabela 3.

Tabela 3 - Modelo de especificação de atividade

Nome da atividade	
Tipo de atividade	Micro / Macro
Atividades relacionadas	
Fluxo da atividade	

Os campos apresentado são apenas um modelo das informações necessárias para a especificação, porém este pode ser instanciado seguindo as necessidades da empresa.

Pessoas Envolvidas: Analista de Negócio
Artefatos de Entrada: BPMN ou Casos de uso de negócio ou Especificação de negócio
Artefatos de Saída: Especificação de atividades
Ferramentas Usadas: Não aplicável

4.1.1.2 Modelagem das micro atividades

Descrição:

Com a utilização do conceito de fluxo de trabalho, diferentemente do processo de desenvolvimento tradicional, é necessário que atividades de baixo nível (computacional) sejam implementadas antes das atividades de alto nível (negócio), isto porque as atividades de negócio utilizaram as atividades computacionais para permitir que o processo modelado seja executado.

Com isto, após a especificação de todas as micro atividades do processo de negócio é necessário a modelagem dessas atividades.

Devido ao fato de que atividades desse tipo estejam relacionadas a elementos computacionais, este passo do processo tem o Desenvolvedor como o principal envolvido, porém o Analista de Negócio tem a importante função de apoiar e validar se as atividades estão sendo modeladas conforme a modelagem do processo de negócio.

Em um processo de desenvolvimento usando técnicas tradicionais, estas atividades seriam implementadas usando linguagens de programação, como Java ou C#, mas neste processo esta atividade é implementada através da interface gráfica, disponibilizada por uma IDE, que permite a modelagem do fluxo de trabalho desta atividade.

Esta atividade é composta por tarefas que realizam atividades computacionais, como – Comunicação com Banco de dados e Execução de serviços.

É importante que todas as atividades criadas sejam nomeadas de forma que o profissional com conhecimento de negócio consiga identifica-las.

O resultado desta atividade é a modelagem de todas as micro atividades do processo de negócio. Essas modelagens são os primeiros elementos da DSL, pois estas serão representadas por elementos gráficos, no momento de modelagem de processo executável.

Pessoas Envolvidas: Profissional de Negócio (20%), Desenvolvedor (80%)
Artefatos de Entrada: Especificação das atividades
Artefatos de Saída: Modelagem (Micro Atividades)
Ferramentas Usadas: IDE de Desenvolvimento com interface gráfica

4.1.1.3 Modelagem das macro atividades

Descrição:

Com os elementos da DSL que representam as micro atividades definidas e a relação entre as duas categorias de atividades é necessário à realização da modelagem das macro atividades.

Essa modelagem é composta por regras de negócio, micro atividades e devido a isso é necessário que para a execução deste passo o empenho dos profissionais técnicos e de negócio, pois estas modelagens realizarão a mistura dos aspectos computacionais e de negócio do processo.

Assim como a modelagem das micro atividades, as modelagens resultantes desta atividade também se tornarão elementos da DSL, porém elementos mais complexos que agrupem outras atividades e que tenham maior ligação com o processo de negócio.

Pessoas Envolvidas: Profissional de Negócio (50%), Desenvolvedor (50%).
Artefatos de Entrada: Especificação das atividades, Modelagem (Micro Atividades).
Artefatos de Saída: DSL Criada (Macro Atividades)
Ferramentas Usadas: IDE de Desenvolvimento com interface gráfica

4.2 Considerações do capítulo

Este capítulo descreveu quais as atividades necessárias para a criação de uma DSL que será aplicada em um projeto de software orientado a serviço. Essas atividades serão exemplificadas no próximo capítulo através da Aplicação da Proposta.

Além disso, o próximo capítulo irá apresentar as possíveis vantagens ou desvantagens de sua utilização aplicando esta proposta em uma aplicação real.

5. APLICAÇÃO DA PROPOSTA

Com o objetivo de analisar as vantagens do uso de uma DSL em um projeto SOA, este capítulo apresenta a implementação de parte de um projeto real através da proposta de processo de criação de DSL apresentado neste trabalho.

5.1 Aprovação de crédito de compra

5.1.1 Contexto de negócio

Uma das maiores distribuidoras de produtos de informática do Brasil, chamada neste trabalho de empresa OD por motivos de sigilo, com o intuito de suprir a necessidade de padronização e otimização de seus processos decidiu a implantação de um sistema de ERP. Esta implantação foi iniciada em meados de junho/2008 e, segundo o último relatório de dezembro/2010, ainda não foi finalizada.

Com esta implantação, surgiu a necessidade de que aplicativos existentes na empresa acessem informações deste novo ambiente. Devido a esta necessidade surgiu o projeto de implantação de um barramento de serviços empresariais (ESB). Através desse barramento, aplicações como o comércio eletrônico ou sistema de vendas poderiam consumir serviços que acessariam informações do ERP.

Entre todos os requisitos do projeto em questão, o experimento é realizado apenas no fluxo de Aprovação de Crédito de Compra. A versão atual do requisito foi desenvolvida por dois Analistas de Sistemas e um Analista de Negócio e teve duração de três semanas.

Este requisito compreende o desenvolvimento de um serviço que orquestra todas as atividades necessárias para solicitação de crédito, por uma revenda, para realização de compra de produtos desta distribuidora.

Esse serviço é iniciado a toda compra realizada pela empresa, eletrônica ou *call-center*, e seu fluxo é descrito pela modelagem de negócio apresentada na figura 16.

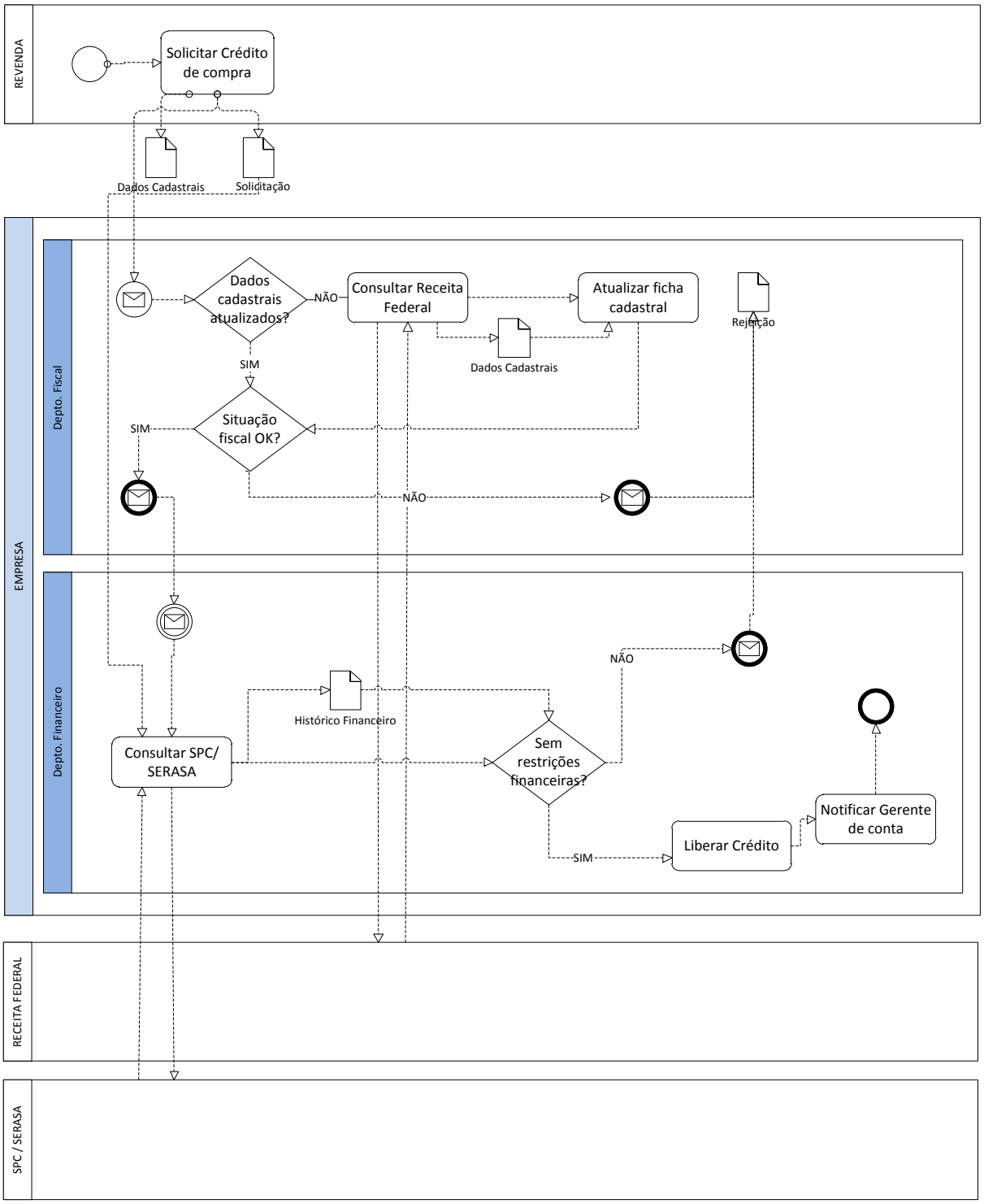

Figura 16 - BPMN Aprovação de crédito e áreas relacionadas

5.1.2 Contexto da aplicação

Atualmente a implementação deste serviço já foi realizada e está em processo de homologação. Essa implementação foi realizada através da derivação

do fluxo de negócio apresentado na Figura 16 para a especificação de casos de uso, realizada por um Analista de Negócio. Após isso, esta especificação serviu como base para o desenvolvimento realizado pelos Analistas de Sistemas. Esses profissionais usaram a linguagem de programação, C#, e devido a isto, o Analista de Negócio não possui controle algum do que realmente foi implementado.

Esse problema ficou evidente quando o processo de homologação do requisito foi iniciado, onde, foi identificado que inúmeros pontos da implementação realizada haviam sido mal interpretados pelos Analistas de Sistemas. Esses problemas causaram grande atraso nos prazos de entrega, fazendo com a duração da fase de testes e correções fosse duas vezes maior que a fase de implementação.

Nessa aplicação foi definida a utilização do workbench de linguagem Workflow Foundation que é incorporado a IDE Microsoft Visual Studio 2010. Com esta escolha não será necessário a criação de interpretadores já que a DSL criada terá a linguagem escolhida como base.

5.2 Aplicação da proposta de processo de criação da DSL

A aplicação da proposta é dividida em dois passos, criação da DSL e sua aplicação. As atividades são executas pelos papéis de Analista de Negócio e Desenvolvedor, porém devido ao número reduzido de recursos para esta aplicação os dois papéis foram desempenhados pela mesma pessoa.

5.2.1 Criação da DSL

5.2.1.1 Identificação das atividades do processo

Descrição:

Esta atividade tem como o objetivo a especificação e categorização das atividades apresentadas no BPMN de Aprovação de Crédito.

Pelo fato de o BPMN apresentado descrever um subprocesso da empresa denominado Aprovação de crédito e a implementação deste subprocesso em um

serviço já ter sido definida, o Analista de Negócio responsável necessita descrever todas as *Tasks* deste BPMN utilizando-se do seu conhecimento do processo da empresa.

A tabela 4 apresenta a especificação da atividade “Liberar Crédito” utilizando o modelo apresentado no capítulo anterior.

Esta atividade foi categorizada como Macro Atividade e possui relacionamento com as atividades “Redução de limite de crédito”, “Geração de boletos” e “Emissão de nota fiscal eletrônica”.

Tabela 4 - Definição da atividade "Liberar Crédito"

Nome da atividade	Liberar Crédito
Tipo de atividade	Macro
Atividades relacionadas	Redução do limite de crédito; Geração de boletos; Emissão de nota fiscal eletrônica.
Fluxo da atividade	
Com a aceitação do crédito esta atividade é iniciada.	
1. Gerar o boleto de todas as parcelas para o pagamento de crédito.	
2. Emitir a nota fiscal eletrônica para a Secretaria da Fazendo.	
3. Notificar a revenda sobre liberação do crédito e da emissão da NF-e.	
4. Subtrair do limite de compra da revenda o valor do crédito liberado.	
5. Enviar boletos para pagamento.	

Pessoas Envolvidas: Analista de Negócio
Artefatos de Entrada: BPMN de Aprovação de Crédito
Artefatos de Saída: Especificação de atividades
Ferramentas Usadas: Microsoft Excel 2010

5.2.1.2 Modelagem das micro atividades

Descrição:

Com a especificação das atividades finalizada, inicia-se o processo de modelagem das atividades em um modelo executável. Para isto, o Desenvolvedor irá separar todas as atividades categorizadas como Micro e modelá-las.

Para a execução desta tarefa foi usando a ferramenta Visual Studio 2010, Microsoft, que disponibiliza a interface gráfica necessária para o desenvolvimento de fluxos de trabalho.

A figura 17 apresenta a modelagem da atividade “Atualização de Ficha Cadastral” que é modelada através de tarefas, que são ações computacionais. Esta atividade é composta de uma tarefa “Atualiza Banco de dados”, que contém duas sub-tarefas – “Envia atualização” e “Recebe Resposta”, que executa um procedimento armazenado em uma base de dados.

Devido ao fato de se tratarem de atividades que em sua maioria tratam de assuntos computacionais, essa modelagem é mantida pela equipe técnica.

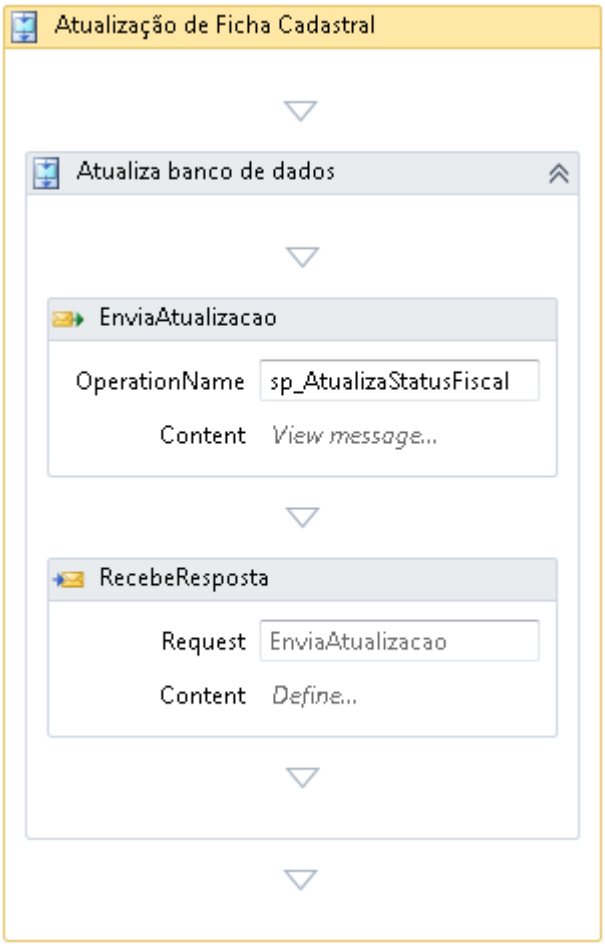

Figura 17 - Atividade "Atualização de Ficha Cadastral"

Pessoas Envolvidas: Analista de Negócio (20%), Desenvolvedor (80%).

Artefatos de Entrada: Especificação das atividades, Modelagem (Micro Atividades).

Artefatos de Saída:

Elementos da DSL criados:

- AtualizacaoFichaCadastral
- ConsultarReceitaFederal
- EnviaEmail
- EmissaoNFe
- GeracaoBoleto
- ConsultaSerasa
- ReducaoLimite

Ferramentas Usadas: Microsoft Visual Studio 2010

5.2.1.3 Modelagem das macro atividades

Descrição:

Após a finalização dos elementos da DSL que representam as micro atividades inicia-se a modelagem das macro atividades.

Esta modelagem foi desenvolvida usando a mesma ferramenta do passo anterior, porém desta vez o Analista de Negócio divide as responsabilidades sobre o modelo com o Desenvolvedor. Isto ocorre, pois essas atividades são compostas de micro atividades, o que permite que aspectos técnicos sejam omitidos.

A figura 18 apresenta um dos modelos resultante. Este modelo representa a atividade “Atualização dados cadastrais” que agrupa logicamente as micro atividades “Consultar Receita Federal” e “Atualizar Ficha Cadastral”.

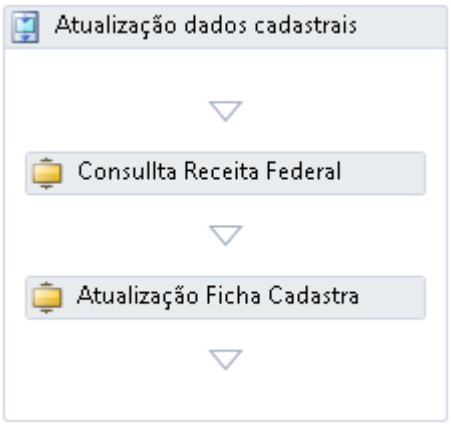

Figura 18 - Atividade "Atualização dados cadastrais"

Na DSL criada foram criados elementos que representam atividades do processo de negócio, estes elementos são representados por um retângulo que contem o nome da atividade. Estes elementos possuem propriedades específicas que indicam as informações necessárias para a execução da atividade, como representado na figura 19, na qual são apresentadas as informações da atividade de “Atualização de dados cadastrais”. Estes elementos serão organizados sequencialmente definindo o fluxo do serviço.

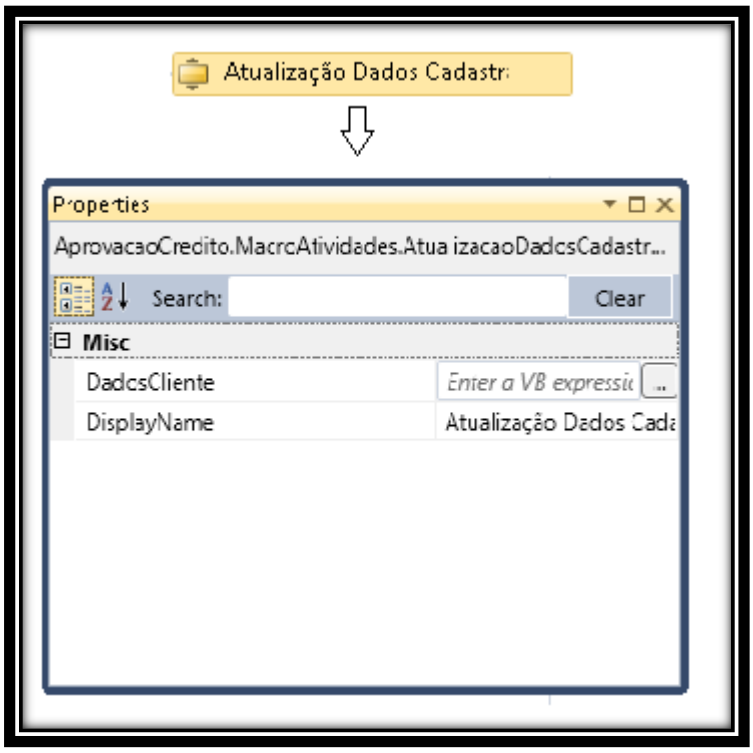

Figura 19 - Informações da atividade “Atualização de dados cadastrais”

Pessoas Envolvidas: Analista de Negócio (50%), Desenvolvedor (50%).
Artefatos de Entrada: Especificação das atividades, Modelagem (Micro Atividades).
Artefatos de Saída:

DSL criada com os elementos:

- AtualizacaoDadosCadastrais
- NoticaRejeicao
- AnaliseCredito
- LiberacaoCredito
- NotificaGerente

Ferramentas Usadas: Microsoft Visual Studio 2010

5.2.2 Aplicação da DSL

5.2.2.1 Definição do processo de negócio executável

Descrição:

Uma vez que a modelagem de todas as atividades está realizada. A DSL está disponível para a criação da modelagem do processo que irá representar processamento do serviço. Esta representação deverá ser criada pelo analista com conhecimento de negócio, com o auxilio de um Analista Técnico, garantindo assim os seguintes aspectos:

- A modelagem estará aderente ao processo de negócio definidos na modelagem de negócio da empresa.
- A terminologia de negócio será utilizada, permitindo que outros analistas sejam capazes de realizar a leitura e entender o modelo.

Para que isto seja possível a IDE usada apresenta uma caixa de ferramenta com todos os elementos da DSL disponíveis. A figura 20 apresenta alguns dos elementos criados pelos passos anteriores, categorizando-os em micro atividades e macro atividades.

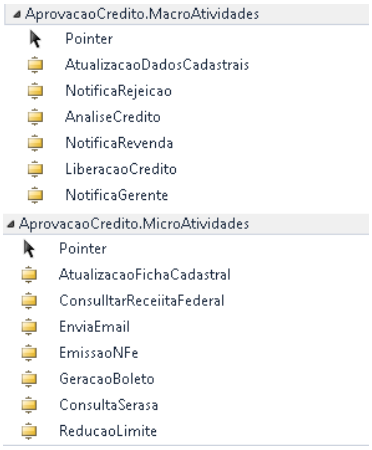

Figura 20 – DSL e seus elementos

Para facilitar a leitura e a quantidade de informações disponíveis na tela, a modelagem foi separada em dois fluxos, Departamento Fiscal e Financeiro. Este último é apresentado na figura 21, e usa dois elementos da DSL criada, *Atualização Ficha Cadastral* e *Notifica Rejeição*, e em seu processo existe a ligação com o fluxo de trabalho do departamento Financeiro.

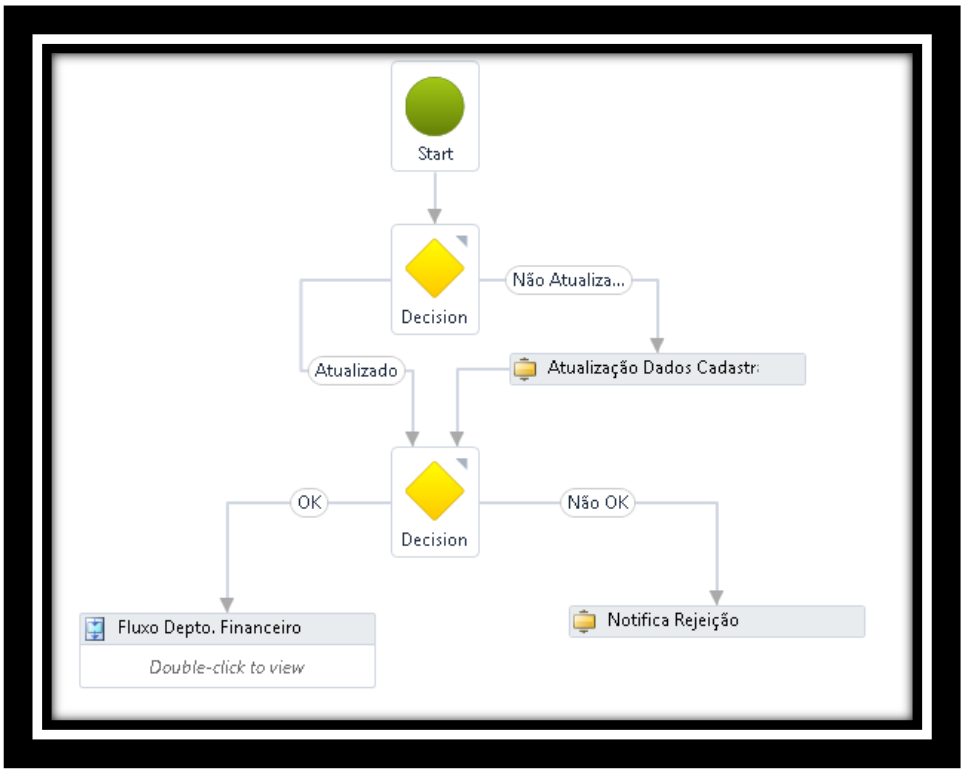

Figura 21 - Modelo executável do fluxo de trabalho do departamento Fiscal

Como pode ser visto na Figura 20 a interface gráfica disponível na IDE permite que o Analista de Negócio, mesmo que com o mínimo conhecimento tecnológico, consiga modelar o fluxo do serviço com pouco auxílio do Desenvolvedor.

Após a finalização desta atividade, a implementação do serviço está finalizada e esta modelagem final será disponibilizada através de um barramento de serviço pela equipe técnica da empresa.

Pessoas Envolvidas: Analista de Negócio (50%), Desenvolvedor (50%).
Artefatos de Entrada: Especificação de atividades, Modelagem (Macro Atividades).
Artefatos de Saída: Processo executável do fluxo de aprovação de crédito
Ferramentas Utilizadas: Microsoft Visual Studio 2010

5.3 Análise de resultados

Com o intuito de analisar os resultados da aplicação da proposta os dados obtidos são comparados com os dados da implementação do mesmo serviço através do processo descrito no item 5.1.2.

A aplicação da proposta de processo permitiu identificar que com a abordagem de utilizar fluxo de trabalho para a modelagem dos elementos da DSL e posteriormente do serviço garante uma maior visibilidade da estrutura do software para o Analista de Negócio, comparando com o processo aplicado originalmente para o desenvolvimento descrito anteriormente. Essa visibilidade permitiu a identificação de possíveis *gaps* ou melhorias no processo de negócio. Outro fator importante é que com o entendimento do Analista de Negócio sobre o software foi possível à criação de testes funcionais e integrados de forma com que partes do processo de negócio sejam testadas.

Uma desvantagem identificada é que normalmente a disponibilidade do Analista de Negócio no processo de desenvolvimento de uma solução é bastante limitada, muitas vezes apenas ocorrendo nas fases iniciais, porém a proposta apresentada demanda que o recurso que irá atuar no papel de Analista de Negócio esteja disponível durante todo o ciclo de desenvolvimento.

Apesar desta desvantagem o envolvimento do Analista de Negócio permitiu que menos erros fossem detectados no processo de validação da solução, o que diminuiu em pelo menos 50% o tempo de desenvolvimento da solução já que na implementação atual 75% do tempo havia sido utilizado para a correção de problemas de implementação. Além do ganho no processo de desenvolvimento, com esta abordagem possivelmente o processo de manutenção da solução compartilhará os mesmos benefícios, porém nenhum experimento foi realizado com o intuito de analisar este aspecto do processo proposto.

Outro aspecto importante é que a proposta de processo idealizava que a atividade de "Definição de processo de negócio executável" fosse executada quase que totalmente pelo Analista de Negócio, porém devido à ferramenta escolhida o Desenvolvedor necessitará oferecer grande apoio nesta atividade, ao menos até a capacitação dos Analistas.

Esta aplicação da proposta mostrou também que com o uso do BPMN como artefato inicial do processo, a sua aplicação se torna menos efetiva, pois a própria modelagem inicial poderia ser transformada para um processo executável. Neste cenário, a única vantagem associada à aplicação do processo seria o ferramental disponível, que muitas vezes não justifica a aplicação de um novo processo.

Em contrapartida é provável que com o uso de casos de uso de negócio como artefato inicial, todos os benefícios da aplicação do processo apresentado anteriormente estariam disponíveis, porém nenhum experimento que comprove essa teoria foi realizado, pois no contexto ao qual o processo foi aplicado era usado BPMN.

O ultimo aspecto identificado foi que com o processo de modelagem tendo como maior responsável o Analista de Negócio, permitiu que os profissionais de TI estivessem atentos apenas a elementos computacionais, como hospedagem, segurança e disponibilidade do serviço.

5.4 Considerações do capítulo

A aplicação da proposta de processo permitiu avaliar, de maneira geral, o processo de criação de DSL e este se mostrou viável e mais efetivo do que o

processo de derivação de código através de casos de uso, utilizado atualmente pela empresa OD, para o desenvolvimento desta solução.

É importante frisar que apesar da escolha do *Windows Workflow Foundation*, realizada nesta aplicação, a proposta de processo apresentada pode ser realizada através de qualquer uma das ferramentas de fluxo de trabalho disponíveis no mercado que possuam a capacidade de geração de código automatizada. A decisão deste produto ocorreu devido a três características:

- Este produto é integrado a IDE Visual Studio, que está entre as melhores e mais produtivas ferramentas disponíveis.
- O uso do Windows Workflow já vem sendo realizada desde o ano de 2006, o que garante uma consolidação no mercado e grande quantidade de fontes de estudo.
- Facilidade de integração devido ao fato de que toda a arquitetura da empresa possui produtos do mesmo fabricante.

Uma desvantagem do uso deste produto é que as informações para a modelagem dos fluxos através da interface gráfica são normalmente conhecidas apenas por profissionais técnicos, porém o processo apresentado possibilitou com que um profissional de negócio consiga realizar a modelagem dos fluxos com o auxílio de um profissional com conhecimento técnico.

Outro importante aspecto do processo de criação é que muitas destas atividades estão presentes na modelagem inicial do Fluxo de Aprovação de Crédito (BPMN), como a atividade “Liberar Crédito” apresentada anteriormente, porém foi possível notar que algumas atividades devem ser obtidas através do entendimento da modelagem e com isto a criação de agrupamentos lógicos se torna necessário, como foi o caso da atividade “Atualização de dados cadastrais”.

6. CONSIDERAÇÕES FINAIS

Este capítulo descreve as conclusões e as contribuições do trabalho e sugestões de trabalhos futuros que poderão ser desenvolvidos a partir deste trabalho.

Com o desenvolvimento do presente trabalho foi possível notar que o objetivo de propor um processo de criação de DSL foi alcançado, porém, apesar de a aplicação do processo ter sido realizada em um projeto real com o intuito de verificar a sua eficácia será necessário que novas aplicações sejam realizadas.

Além disso, o presente trabalho apresentou um estudo detalhado da definição de DSL, algo que apesar de existir a muitos anos ainda não está consolidado. Este trabalho pode ser usado como incentivo ao uso de DSL no processo de desenvolvimento de software.

6.1 Contribuições do Trabalho

Alguns trabalhos relacionados foram analisados e estes apresentaram a importância e o valor para o uso das DSL no processo de desenvolvimento de software. Outro ponto a se destacar é que sendo os trabalhos apresentados desenvolvidos recentemente este trabalho propõe uma DSL.

Além dessas contribuições este trabalho tem como principal contribuição o processo de criação de DSL. Este processo de criação foi aplicado e comparado a um projeto real e apesar do mesmo já fazer uso de BPMN, apresentou inúmeras melhorias comparadas ao processo original.

6.2 Trabalhos Futuros

Apesar do processo se mostrar eficaz no cenário apresentado com o uso de BPMN, é possível que novas aplicações sejam realizadas com o intuito de validar o

processo de criação proposto, como no caso da utilização de casos de uso de negócio.

Outro trabalho poderia ser a constatação das melhorias apresentadas no processo de manutenção devido ao uso da DSL e da capacidade de entendimento do profissional de negócio na solução desenvolvida.

Além desses trabalhos que dariam subsídios à eficiência do processo proposto, um novo trabalho possível é a aplicação ou adaptação do processo de criação de DSL fora do contexto de SOA.

REFERÊNCIAS

CARDENAS, A. et al. **Can Domain-Specific Languages Be Implemented by Service-Oriented Architecture?** SAC'10. Sierre: ACM. 2010.

ERL, T. **Service-Oriented Architecture:** Concepts, Technology, and Design. [S.l.]: Prentice Hall PTR, v. I, 2005. 792 p.

FOWLER, M. Language Workbenches: The Killer-App for Domain Specific Languages? **Marting Fowler**, 2005. Disponivel em: <http://martinfowler.com/articles/languageWorkbench.html>. Acesso em: 30 nov. 2010.

FOWLER, M. DslQandA. **Martin Fowler**, 2008. Disponivel em: <http://www.martinfowler.com/bliki/DslQandA.html>. Acesso em: 18 March 2011.

FUGITA, H. S. **MAPOS:** Método de Análise e Projeto Orientado a Serviços. Dissertação (Mestrado). São Paulo: Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais. 2009.

LINDBERG, O.; THORIN, P. **Evaluating Microsoft Domain-Specific Language Tools – an Empirical Study of Domain-Specific Languages and Service-Oriented Architecture**. In 7th Conference on Software Engineering Research and Practice (SERPS'07). Göteborg: IT University of Göteborg. 2007. p. 61-68.

MICROSOFT. Context for XPath Expressions. **MSDN Library**, 2010. Disponivel em: <http://msdn.microsoft.com/en-us/library/ms256199.aspx>. Acesso em: 15 nov. 2010.

MOTAL, T.; ZAPLETAL, M.; WERTHNER, H. **The Business Choreography Language (BCL) - a Domain-Specific Language for Global Choreographies**. World Conference on Services - II. Bangalore: IEEE Congress on. 2009. p. 150-159.

OASIS OPEN. Reference Model for Service Oriented Architecture 1.0. **OASIS Standard**, 2006.

OBERORTNER, E.; ZDUN, U.; DUSTDAR, S. Domain-specific Languages for Service-oriented Architectures: An Explorative Study, Vienna, 2008.

PAPAZOGLOU, M. P. **Service -Oriented Computing:** Concepts, Characteristics and Directions. In Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE). Roma: IEEE Computer Society. 2003. p. 3-12.

PIETRO-DIAZ, R. **Domain Analisys for reusability**. COMPSAC'87. Tokyo: [s.n.]. 1987. p. 23-29.

RAILS. Getting Started with Rails. **Rails Guide**, 2010. Disponivel em: <http://guides.rubyonrails.org/getting_started.html>. Acesso em: 25 jan. 2011.

ROSA, A.; AMARAL, V.; BARROCA, B. **Use of MS DSL Tools in the development process of a Domain-Specific Language**. INForum 2009. Lisboa: Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa. 2009. p. 515-526.

W3C. XSL Transformations (XSLT) Version 1.0. **W3C Recommendation**, 1999. Disponivel em: <http://www.w3.org/TR/xslt>. Acesso em: 15 nov. 2010.

W3C. Sobre o Consórcio W3C. **W3C Brasil**, 2008. Disponivel em: <http://www.w3c.br/sobre/>. Acesso em: 15 nov. 2010.

W3C. Schema. **W3C**, 2009. Disponivel em: <http://www.w3.org/standards/xml/schema>. Acesso em: 15 nov. 2010.

W3C. Extensible Markup Language (XML). **W3C**, 2010a. Disponivel em: <http://www.w3.org/XML/>. Acesso em: 01 25 2011.