UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

Relatórios Técnicos
do Departamento de Informática Aplicada
da UNIRIO
0023/2009

Metodologia para Análise e Projeto de Serviços em uma abordagem SOA

Jairo Francisco de Souza
Leonardo Guerreiro Azevedo
Fernanda Baião
Flávia Santoro

Departamento de Informática Aplicada

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

Projeto de Pesquisa

Grupo de Pesquisa Participante

Patrocínio

PETROBRAS

Relatórios Técnicos do DIA/UNIRIO, No. 0023/2009
Editor: Prof. Sean W. M. Siqueira Outubro, 2009

Metodologia para Análise e Projeto de Serviços em uma abordagem SOA*

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

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

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

Abstract. In order to enhance service discovery, services must be well described and be published in a registry. Services based applications are developed as a set of independent services with low-coupling and well-defined interfaces which are used by their consumers. Traditional methods, such as "object oriented" and "component based" ones, are not properly for service development, mainly in analysis and design aspects. Hence it is a challenge in SOA to propose a method for service analysis and design that fits enterprise needs. The goal of this work is to propose such methodology. In order to produce services aligned with business, the proposal has as input service candidates identified from business process models.

Keywords: SOA, Web Services Analysis and Design.

Resumo. Serviços devem ter descrição bem definida e serem publicados em um repositório, permitindo serem facilmente localizados. Aplicações baseadas em serviços são desenvolvidas como conjuntos independentes de serviços oferecendo interfaces bem definidas para seus usuários potenciais utilizando o princípio de baixo acoplamento. Métodos tradicionais, tais como "orientado a objeto" e "baseado em componentes" especialmente nos aspectos de análise e projeto são inadequados para os desafios e características necessárias para a construção de serviços. Portanto, um importante desafio em SOA é a definição de uma metodologia de análise e projeto de serviços que seja adequada às características da organização. Este trabalho tem o objetivo de propor uma metodologia para análise e projeto de serviços considerando como base a identificação de serviços a partir da modelagem de processos de negócio.

Palavras-chave: SOA, Análise e Projeto de Serviços.

* Trabalho patrocinado pela Petrobras.

Sumário

1 Introdução

Neste relatório são apresentadas abordagens para análise e projeto de serviços. Para definição da proposta, os conceitos e princípios relacionados a serviços em SOA devem ser bem entendidos. Em [Azevedo *et al.,* 2008a] são apresentados os principais conceitos relacionados à SOA. Além disso, a metodologia proposta leva em consideração o método de identificação de serviços a partir da modelagem de processos de negócio proposto em [Azevedo *et al.,* 2009a; Azevedo *et al.,* 2009b; Azevedo *et al.,* 2008b].

[Papazoglou, 2007] define que serviços de software são elementos computacionais auto-contidos, independentes de plataforma que suportam composição fácil, rápida e de baixo custo de aplicações distribuídas mantendo baixo acoplamento.

Serviços devem ter descrição bem definida e serem publicados em um repositório, permitindo serem facilmente localizados. Além disso, podem ser compostos para criar sistemas baseados em serviços e aplicações complexos. Serviços ajudam a integrar aplicações que não foram escritas com a intenção de serem integradas com outras aplicações e definem arquitetura para construir novas funcionalidades integrando funcionalidades de aplicações existentes.

Jamshidi *et al.* [2009], corrobora Arsanjani [2004], Zimmermann *et al.* [2004a] e Arsanjani *et al.* [2008], ressaltando que métodos tradicionais, tais como "orientado a objeto" e "baseado em componentes"especialmente nos aspectos de análise e projeto são inadequados para os desafios e características necessárias para a construção de serviços. Arsanjani [2004a] apresenta que a modelagem de serviços vai além das práticas OOAD (*Object-Oriented Analysis and Design*), uma vez que a abordagem OOAD não leva em consideração elementos chaves para abordagens SOA como os requisitos do consumidor e provedor do serviço, os objetivos empresariais, questões de governança, etc.

[Papazoglou, 2008] enfatiza que os métodos tradicionais de desenvolvimento não tratam dos três elementos chaves de uma SOA, a saber: serviços, composição de serviços e serviços percebidos como componentes. Além disso, ele ressalta que as atividades mais recentes de pesquisa concentram em como prover princípios e guias para especificar, construir e refinar e customizar processos de negócio a partir de um conjunto de serviços internos e externos.

Jamshidi *et al.* [2009] apresenta que a especificação de serviços é o ponto principal da atividade de modelagem orientada a serviços e foca na elaboração do projeto detalhado de serviços, fluxos e componentes. Portanto, a seleção de um método apropriado e comprovado para conduzir esta atividade tem caráter crítico para o sucesso de uma solução baseada em serviços.

Uma abordagem para modelagem de serviços deve tratar, no mínimo, as quatro partes fundamentais de serviços [Papazoglou, 2008]:

- Estrutura: envolve a definição do tipo do serviço, mensagem, interfaces e operadores (assinatura serviço);
- Comportamento: corresponde à documentação do entendimento dos efeitos do serviço e suas operações, detalhamento da semântica das mensagens de

entrada e saída, por exemplo, em um serviço de "Cadastro de pedidos", "Como podemos cancelar ou atualizar um pedido?".

- Política: descrições das assertivas relacionadas às políticas e restrições do serviço, considerações de controle de qualidade sobre os serviços em relação às partes que interagem com ele. O serviço prescreve, limita, ou especifica qualquer aspecto de contrato do negócio (SLA – *Service Level Agreement*).
- Vocabulário e melhores práticas: inclui definição de:
    - Processos de negócio comuns a fim de alcançar canonização/padronização de processos e serviços;
    - Formatos de intercâmbio de dados comuns, por exemplo, mensagens/protocolos padrão que são trocados no contexto de um processo/transação;
    - Terminologia comum no nível de itens de dados e mensagens a fim de acomodar diferentes terminologias de serviços.

1.1 Metodologia

A metodologia de trabalho foi dividida nos seguintes passos:

1. Levantamento bibliográfico.
2. Leitura e elaboração de resumos dos artigos mais interessantes ao projeto.
3. Discussões em reunião dos artigos estudados, a partir de apresentações dos trabalhos.
4. Elaboração de relatório dos estudos realizados.
5. Definição de heurísticas para análise e projeto de serviços.

Este relatório foi produzido pelo Projeto de Pesquisa em SOA como parte das iniciativas dentro do contexto do Projeto de Pesquisa do Termo de Cooperação entre NP2Tec/UNIRIO e a Petrobras/TIC-E&P/GDIEP.

1.2 Estrutura do relatório

Esse relatório está organizado em 4 capítulos, sendo o capítulo 1 a presente introdução.

No capítulo 2 são apresentados os trabalhos estudados.

Nos capítulos 3 e 4 são apresentadas propostas de metodologias para análise e projeto de serviços, respectivamente.

Nos capítulos 5, 6 e 7 são apresentadas as conclusões do trabalho, referências bibliográficas e anexos do trabalho.

2 Trabalhos relacionados

Como parte do trabalho de definição de uma abordagem para análise e projeto de serviços, foram estudados diferentes trabalhos encontrados na literatura sobre o

assunto. Este capítulo apresenta um resumo destes trabalhos relacionados avaliados. Uma descrição mais detalhada dos trabalhos está presente no Anexo I.

Metodologias para análise e projeto de serviços são propostas na literatura [Arsanjani, 2004; Endrei, 2004; Zimmermann *et al.*, 2004a; Zimmermann *et al.*, 2004b]. Além das metodologias para projeto de serviços, foram encontradas também diferentes abordagens relacionadas com a identificação de serviços [Adam *et al.*, 2008; Fareghzadeh, 2008; Jamshidi *et al.*, 2008; Jamshidi *et al.*, 2009; Kohlborn *et al.*, 2009]. Contudo, as metodologias e abordagens analisadas não são especificadas o suficiente para serem automatizadas e operacionalizadas, os quais são os objetivos desse trabalho. Kohlborn *et al.* [2009] avaliaram 30 metodologias para identificação e análise de serviços. Com resultado da avaliação, eles concluem que, apesar da orientação a serviços estar afetando diferentes níveis de uma organização, nenhuma maneira unificada de identificar e analisar serviços foi identificada. Eles analisaram abordagens elaboradas pela academia e pela indústria e observaram que o escopo e nível de profundidade das abordagens variam muito. Segundo eles, nenhuma das abordagens examinadas é compreensível e integrada o suficiente para cobrir de maneira satisfatória os principais conceitos de serviços (serviços de negócio e serviços de software), como também apontado por Kohlmann e Alt [2007] e Klose *et al.* [2007]. [Sanz *et al.*, 2009] definem que serviços de negócio correspondem a ações específicas que são realizadas por uma organização. Kohlborn *et al.* [2009] definem serviços de software como sendo partes de uma aplicação os quais podem ser consumidos separadamente por diferentes entidades. Um serviço de negócio expõe funcionalidades de aplicações que podem ser reutilizadas e compostas baseado nas necessidades do negócio. Portanto, um serviço de software apóia a execução de um serviço de negócio. A principal tecnologia para implementação de serviços de software é *web services* [Alonso, 2004]. Esta definição difere da definição proposta em [Azevedo *et al.*, 2009a, 2009b], a qual define serviço de negócio como sendo uma especialização de serviço candidato que trata lógica do negócio. Neste trabalho, chamaremos este tipo de serviço candidato como sendo serviço de lógica.

Kohlborn *et al.* [2009] apontam que, como conseqüência da proliferação da idéia dos níveis de serviços de negócio e serviços de software, agora existe uma demanda por metodologias de engenharia de serviços que cubram ambos serviços de negócio e serviços de software e provejam uma abordagem holística e integrada para garantir alinhamento entre negócio e TI. Em [Azevedo *et al.*, 2009] foi proposta a identificação de serviços candidatos a partir de processos de negócio. Neste relatório pretende-se definir heurísticas de análise e projeto que permitam definir os serviços de software a partir dos serviços candidatos.

As propostas apresentadas nesta seção proporcionaram idéias que foram incorporadas na nossa proposta.

A seção 2.1 apresenta um resumo sobre as metodologias para análise e projeto de serviços, juntamente com uma proposta de especificação de serviços em UML. A seção 2.2 apresenta um resumo sobre as abordagens de identificação de serviços.

2.1 Metodologias para análise e projeto de serviços

Arsanjani [2004] propõe SOMA (*Service-Oriented Modeling and Architecture*) que corresponde a uma metodologia para implantação de abordagem SOA em uma organização, composta de um conjunto de camadas para SOA. Apesar de Arsanjani

[2004] apresentar um conjunto de passos para modelagem e arquitetura de serviço, as atividades não são apresentadas em detalhes, sendo necessário um maior aprofundamento das mesmas. Além disso, a metodologia deve ser instanciada para cada projeto (onde decisões específicas de projeto devem ser tomadas). Porém, são poucos os indicativos que o autor fornece para basear decisões de projeto.

Zimmermann *et al.* [2004a] apresentam uma proposta para combinação de elementos de práticas bem estabelecidas, tais como OOAD (*Object Oriented Analysis and Design*), EA (*Enterprise Architecture*), e BPM (*Business Process Modeling*), complementando-os com elementos específicos da nova demanda. Sua proposta de metodologia para análise e projeto de serviços em SOA é baseada em um conjunto de critérios de qualidade e requisitos desejados. A partir de elementos adequados de OOAD, EA, e BPM, é definida uma abordagem híbrida *Service-Oriented Analysis and Design* (SOAD), combinando elementos destas disciplinas. A abordagem incorpora novos elementos, tais como conceituação de serviço (identificação), agregação e categorização de serviços, políticas, processo *meet-in-the-middle*, semântica, e discriminação de serviços (para reutilização). Os requisitos apresentados para a metodologia estão em conformidade com a nossa proposta. No entanto, não descrevem as soluções para as questões levantadas (que são bastante semelhantes àquelas levantadas no nosso projeto).

Endrei [2004] tem como objetivo introduzir e descrever padrões para comércio eletrônico (*e-business patterns*) desenvolvidos pela IBM. Além disso, apresenta diversos conceitos de SOA e Serviços Web, indica algumas boas práticas na área e descreve tecnologias da IBM. O trabalho é bastante completo, descreve em detalhes e de forma clara quase todos os conceitos e tecnologias relacionados a SOA e Web Services. Pode ser usado como referência para entendimento e aprendizagem. Além disso, apresenta passos para uma metodologia voltada para SOA que incorpora o uso dos padrões propostos e exemplifica quando e como utilizá-los. O ponto negativo (para o projeto) é que não detalha ou apresenta heurísticas específicas para a questão da granularidade de serviços. No entanto, aponta uma lista de boas práticas que poderia ser estendida, ou mais bem trabalhada, para ajudar a gerar as heurísticas desejadas.

Por sua vez, Simon Johnston [2005a] apresenta um *profile* UML criado pela IBM para descrição de serviços nas ferramentas Rational. O *profile* UML, para descrição de serviços, criado pela IBM Rational é utilizado para descrever atividades de desenvolvimento do ciclo de vida do serviço e visões para diferentes *stakeholders*.

2.2 Abordagens para identificação de serviços

Fareghzadeh [2008] divide sua proposta de identificação de serviços em Análise Inicial, Análise em Profundidade e Criação de uma taxonomia de serviços. O autor trata a utilização de vários modelos como insumo para a análise de serviços candidatos e também propõe lidar com dois mundos comumente distintos: as demandas da TI e as demandas do negócio. Contudo, o método para identificação de serviços candidatos apresentado é fortemente dependente da avaliação de um analista, o que apresenta grandes dificuldades para automatizar o processo. O artigo não apresenta princípios que poderiam ser utilizados na análise do serviço candidato identificado a fim de ajudar a decidir se este deve ser implementado.

O objetivo do artigo [Jamshidi *et al*., 2008] é tratar aspectos dos passos iniciais para construção de soluções baseadas em serviço – modelagem de serviços, considerando modelos de negócio. O artigo propõe um novo processo adequado para identificação e especificação de serviços e de seus elementos arquiteturais a partir de modelos de negócio. A proposta busca satisfazer métricas de desempenho gerencial e técnica. Como positivo, o artigo apresenta conceitos, princípios, elementos de modelo e diretrizes para a construção de soluções orientadas a serviços a partir da modelagem de processos de negócio. Deve-se enfatizar o passo 2 (de identificação de serviços utilizando uma metodologia de agrupamento) e o passo 4 (de documentação dos serviços). O passo 1 corresponde à modelagem de processos de negócio e o passo 3 corresponde à decisão se o serviço vai ser implementado como um artefato de um BPMS ou como um serviço construindo em uma arquitetura SOA. Como ponto negativo está o fato do método basear-se em EBP (processos de negócio elementares) que é algo subjetivo para ser alcançado. Além disso, a proposta trata todos os serviços como sendo do mesmo tipo e identificados a partir de EB (Entidades do Negócio), não havendo separação entre serviços de lógica e serviços de dados, por exemplo. Além disso, os autores propõem uma maneira automatizada para geração de artefatos para a especificação de serviços. No entanto, a proposta é apresentada de forma genérica e sem um exemplo prático, o que dificulta o entendimento e instanciação da proposta na prática (o que não é também apresentado no artigo).

Adam *et al.,* [2008] tratam o desafio de identificar serviços no nível adequado de abstração e com um conjunto de funções correto. Conjunto de funções refere-se à quantidade de operações diferentes providas pelo serviço, enquanto que nível de abstração descreve se uma operação realiza uma funcionalidade mais orientada ao negócio ou uma funcionalidade mais técnica. O artigo apresenta um método para derivar sistematicamente serviços (*web services*), baseado nos processos de negócio de um conjunto representativo de clientes de uma organização. Dessa forma, o artigo tem o objetivo de identificar serviços entre organizações, e não dentro da própria organização. Os autores definem atividade elementar como sendo uma atividade descrita em mais detalhes, por exemplo, a descrição da atividade inclui como um papel deve interagir com um sistema. A proposta é que uma atividade elementar seja descrita usando um *template* de caso de uso mínimo [Cockburn, 2001] a fim de explicitar todos os passos de interação alternados executados para realizá-la. Este detalhamento muitas das vezes não é o objetivo da modelagem de processos de negócio. A nossa proposta identifica serviços candidatos em diferentes granularidades, não requerendo o detalhamento do modelo de processos de negócio a nível conceitual. Na metodologia de identificação de serviços proposta neste projeto [Azevedo *et al.*, 2009], inicialmente, os parâmetros de entrada e saída do serviço são obtidos a partir de uma das atividades em que o serviço está inserido. Na nossa proposta, como consideramos requisitos e regras de negócio, temos um nível de granularidade tal que conseguimos ter os parâmetros de entrada e saída do serviço mais explícitos e mais genéricos. O mesmo não ocorre no caso de serviços identificados a partir de AND, XOR etc.

O objetivo do artigo [Kohlborn *et al.,* 2009] é apresentar um estudo e análise feitos sobre diversos métodos para identificação de serviços, verificando seus pontos fortes e fracos (com base em critérios definidos: Conceito de SOA, Estratégia de implementação de SOA, Cobertura do ciclo de vida, Grau de prescrição, Acessibilidade e validade). A partir do resultado obtido, propõe um novo método que contemple todos os requisitos considerados. O método proposto trata de várias

questões relevantes para identificação de serviços, reutilizando propostas da literatura. A proposta final parece ser um bom guia, no entanto, não apresenta um estudo de caso, apenas um exemplo de uso.

3 Proposta de metodologia para análise

A fase de análise de serviços candidatos corresponde à fase seguinte à identificação de serviços candidatos e após ela segue-se para fase de projeto de serviços físicos. A Figura 1 apresenta um modelo destas três fases em macro-processos.

Figura 1 – Primeiras 3 fases do desenvolvimento de serviços candidatos

Portanto, a fase de análise de serviços candidatos tem como entrada os serviços candidatos que foram identificados na fase *Identificar serviços candidatos*, utilizando a metodologia proposta em [Azevedo *et al.*, 2008b, 2009a, 2009b], cujos passos são apresentados na Figura 2. A fase de análise produz como saída um conjunto de serviços na granularidade mais adequada para serem projetados para implementação (fase seguinte da metodologia). Esta fase é composta pelas seguintes etapas (Figura 3):

1. Priorizar serviços candidatos
2. Definir granularidade inicial de serviços candidatos
3. Agrupar serviços candidatos de dados
4. Modelar serviços candidatos

Nas seções a seguir estas são etapas apresentadas em maiores detalhes, onde são propostas heurísticas para executá-las. Também são apresentados exemplos de uso das heurísticas utilizando para um modelo de processos de negócio de uma grande empresa brasileira.

Figura 2 – Detalhamento da fase Identificar serviços candidatos: Metodologia para identificação de serviços candidatos [Azevedo *et al.*, 2009a]

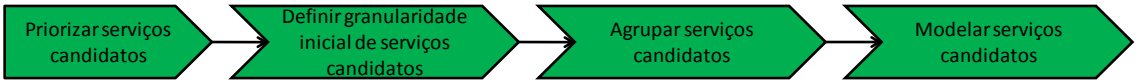


Figura 3 – Detalhamento da fase Analisar serviços candidatos

3.1 Priorização de serviços candidatos

A primeira etapa da análise de serviços corresponde à priorização de serviços candidatos. O objetivo desta etapa é entender quais serviços candidatos mais contribuem para a organização e devem ser considerados inicialmente para análise.

Kolhborn *et al.* [2009] apontam que estas informações são fundamentais para tomar decisões de desenvolvimento dos serviços ou aquisição/uso de serviços fornecidos por outras organizações (*sourcing*) e uso de recursos financeiros. Eles propõem a análise de cadeias de valores ou redes do negócio a fim de permitir o entendimento das funções críticas bem como funções de apoio da organização. Esta análise é aplicada sobre serviços de negócio a fim de priorizá-los. Sanz *et al.* [2006] definem serviço de negócio com sendo um conjunto específico de ações que são realizadas por uma organização. Nesta definição, um serviço de negócio tem um nível de abstração tal que independe da ação estar automatizada ou sendo realizada manualmente. A definição caracteriza uma ação que o negócio trata. Esta definição difere da definição proposta em [Azevedo *et al.*, 2009a, 2009b], a qual define serviço de negócio como sendo uma especialização de serviço candidato que trata lógicas do negócio, ou seja, serviço de negócio implementa uma lógica do negócio identificada em um serviço candidato (o qual não inclui ações realizadas manualmente).

Neste relatório, a priorização está sendo aplicada em serviços candidatos os quais, apesar de ainda não corresponderem a serviços físicos, já não são mais serviços tão abstratos como os serviços de negócio propostos por Sanz *et al.* [2006]. No caso de serviços candidatos, não se está analisando toda a cadeia de valores da organização (como sugerido por Kolhborn [2009]), mas sim um ou poucos processos de negócio que foram definidos previamente. Logo, não se têm vários processos em uma cadeia de valor com diferentes prioridades e os serviços candidatos quanto a este critério teriam, em geral, o mesmo peso. Marks e Bell [2006] propõem a elaboração de uma matriz de priorização de acordo com as diretrizes da organização enquanto que Jones [2006] propõe o uso de uma matriz de ordenação de serviços para ordenação de serviços *ad hoc* baseada no valor que o serviço disponibiliza. Estas propostas de priorização são muito subjetivas, e argumentamos que é importante se ter um critério mais objetivo de priorização. Saaty [1990] propõe uma metodologia estruturada chamada de AHP (*Analytical Hierarchy Process*) baseada no princípio de decomposição de um problema complexo em problemas menores que podem ser resolvidos mais facilmente e explicitamente levando em consideração dados financeiros e não-financeiros. Hafezz *et al.* [2002] propõem o uso de AHP para determinar as capacidades chaves de uma organização, auxiliando na priorização de serviços. Kholborn [2009] apontam que apesar de AHP usar uma metodologia estruturada e formal, os critérios definidos são puramente subjetivos baseados nos julgamentos feitos por pessoas responsáveis pela tomada de decisões. Logo, argumentamos a necessidade de definir medidas mais objetivas para priorização. Neste trabalho, propomos a priorização de serviços candidatos seja feita a partir da análise dos elementos presentes nos processos de negócio feita de forma automatizada, diminuindo o máximo possível a subjetividade desta etapa.

Neste trabalho propomos o uso das seguintes informações produzidas na etapa de identificação de serviços candidatos: grau de reuso de cada serviço candidato, se o serviço foi identificado a partir de atividades de múltiplas instâncias, associação de serviços candidatos com os sistemas que implementam estes requisitos, associação dos serviços candidatos com os requisitos da demanda e associação de serviços candidatos com papéis. A equipe deve definir pesos para cada uma destas informações e ordenar os serviços de acordo com estes pesos. A seguir, seguem sugestões para uso de cada uma destas informações.

3.1.1 Grau de reuso de cada serviço candidato

Heurística: Agrupar serviços candidatos para cada diferente grau de reuso encontrado, listando grau de reuso e quantidade de serviços candidatos correspondente ao grau de reuso. Em seguida, definir pesos para cada agrupamento.

A Tabela 1 (colunas *Grau de reuso* e *Quantidade de serviços* candidatos) apresenta o agrupamento de serviços candidatos resultante da execução do método de identificação de serviços em um exemplo de modelo de processos de uma grande empresa brasileira. Esta tabela apresenta os diferentes graus de reuso para os 110 serviços candidatos identificados: grau de reuso igual a 1 significa que o serviço só é usado em um único lugar e igual a 8 significa que o serviço é usado em oito lugares diferentes. Quanto maior o grau de reuso de um serviço candidato maior o ganho com a disponibilização do serviço candidato como um serviço implementado (por exemplo, como um *web service*). Um serviço candidato com grau de reuso igual a 1 não traz ganho para a organização em ser disponibilizado como um serviço físico. Logo, o peso para este serviço seria igual a zero. No entanto, um serviço com grau de reuso igual a 8 traz grande ganho para a organização em disponibilizá-lo como um serviço físico. Dessa forma, considerando pesos entre 1 a 5, uma proposta de pesos seria atribuir peso 5 para os serviços com grau de reuso igual a 7 e 8, peso 3 para serviços com grau de reuso igual a 4 e 3, peso 2 para serviços com grau de reuso 2 e peso 0 para os serviços com grau de reuso igual a 1, como apresentado na Tabela 1, coluna "Pesos". Em uma análise mais ampla, pode-se definir pesos para grau de reuso a serem aplicados como padrão em diferentes projetos de desenvolvimento de serviços. Estes pesos podem ser definidos a partir de uma análise da aplicação do método em diferentes projetos.

Tabela 1 – Exemplo de agrupamentos por grau de reuso de serviços candidatos

Grau de reuso	Quantidade de serviços candidatos	Pesos
1	77	0
2	9	1
3	17	3
4	1	3
7	2	5
8	1	5

3.1.2 Associação de serviços candidatos com sistemas

Heurística: Agrupar serviços candidatos que estão associados a cada sistema que os implementa, listando sistema e quantidade de serviços candidatos. Em seguida, definir pesos para cada agrupamento.

Neste caso a definição do peso depende muito mais da importância do sistema para a organização do que da quantidade de serviços identificada para o sistema. Além disso, o serviço candidato deve ser marcado como implementado quando da execução do método de identificação de serviços, indicando que o elemento do processo de negócio do qual o serviço candidato foi identificado está implementado em um sistema.

No caso do processo utilizado neste trabalho, dois sistemas são considerados Apl1 e Apl2. A Tabela 2 apresenta uma sugestão de pesos para os diferentes sistemas. Observe que podem existir serviços candidatos associados a mais de um sistema, por exemplo, no caso do processo do processo utilizado existe 1 serviço associado a Apl1 e a Apl2 ao mesmo tempo. Neste caso, pode-se considerar como peso o somatório do peso atribuído a cada sistema.

Também há o caso de serviços candidatos não aparecerem associados a nenhum sistema após a execução do método de identificação de serviços. Isto pode ocorrer porque realmente não existe um sistema implementando a funcionalidade exposta pelo serviço candidato ou por não existir no modelo de processos a associação explícita do elemento do qual o serviço candidato foi identificado com o sistema que o implementa. Dessa forma, é importante verificar se algum serviço poderia ser associado manualmente a um sistema e, depois de realizar esta análise, definir qual seria o melhor peso a ser atribuído para os serviços que não estão associados a nenhum sistema, justificando o preso atribuído. No caso do processo utilizado para elaborar os exemplos, existem 11 serviços candidatos que não estão associados a nenhum sistema, para este caso propõe-se atribuir peso 1.

Tabela 2 - Exemplo de agrupamentos por sistemas que implementam serviços candidatos

Sistema	Quantidade de serviços candidatos	Pesos
Apl1	14	2
Apl2	84	3
Apl1, Apl2	1	5
(Nenhum)	11	1

Heurística: Indicar serviços candidatos que poderão potencialmente ser invocados por sistemas.

Seguindo ainda o princípio do reuso em SOA, o próximo passo é identificar o potencial de reuso de serviços candidatos por sistemas, inclusive outros sistemas além dos já explicitados. Logo, neste caso, a análise deve incluir também os serviços que não estão sendo utilizados por nenhum sistema a fim de identificar qual sistema poderia passar a utilizá-los. Além disso, de acordo com o objetivo do projeto, é importante fazer uma análise mais ampla, incluindo também outros potenciais sistemas que poderão utilizar os serviços, além dos que participam dos modelos de processos analisados. A identificação de outros sistemas pode ser feita a partir de modelos de processos cujo contexto está relacionado aos modelos de processos sendo analisados, ou através da análise de documentação dos sistemas existentes, para os casos em que não existe modelo de processos. A Tabela 3 apresenta um exemplo de serviços possivelmente reutilizados por outros sistemas. Observe que existem serviços reutilizados tanto pelo Apl1 como pelo Apl2, que são os sistemas que estão sendo utilizados no modelo de processos. Além disso, o sistema Apl3 também foi listado como um sistema que possivelmente irá reutilizar os serviços candidatos identificados no processo. Logo, durante a fase de projeto, deve-se analisar como este sistema poderá utilizar estes serviços a fim de generalizá-los para este fim.

Tabela 3 - Exemplo de agrupamentos por sistemas que possivelmente implementam serviços candidatos

Sistema	Quantidade de serviços candidatos possivelmente utilizados	Pesos
Apl1	22	4
Apl2	93	4
Apl3	5	2

Observe que, caso exista apenas um sistema associado às atividades do processo, então todos os serviços candidatos teriam o mesmo peso, diferindo apenas dos serviços candidatos que não estão associados a nenhum sistema. Portanto, podem-se definir pesos para estes dois casos, ou não executar esta heurística, caso a informação do sistema associado não seja relevante para a análise.

3.1.3 Aumentar peso de serviços candidatos identificados a partir de atividades de múltiplas instâncias

Heurística: Atribuir maior peso para serviços candidatos identificados a partir de atividades de múltiplas instâncias.

Outra questão importante é que serviços identificados por atividades de múltiplas instâncias tendem a ter um uso muito maior do que serviços identificados a partir de atividades que não são de múltiplas instâncias. Logo, estes serviços devem ter um maior peso. Logo, sugere-se incrementar de 1 (ou outro valor definido) a prioridade de serviços identificados a partir de atividades de múltiplas instâncias.

3.1.4 Associação dos serviços candidatos com os requisitos da demanda

Heurística: Definir pesos para serviços candidatos associados aos requisitos da demanda. Definir maior peso para serviços candidatos associados aos requisitos de uma demanda com maior prioridade.

A identificação de serviços candidatos a partir da modelagem de processos de negócio pode levar à identificação de serviços candidatos que não necessariamente estão diretamente relacionados aos requisitos da demanda, pois a demanda pode estar tratando a disponibilização de apenas parte do modelo de processos sendo analisado. Dessa forma, pesos diferentes devem ser atribuídos para cada um destes casos. No caso do processo analisado neste trabalho, todos os serviços candidatos estão associados à demanda. Logo, todos os serviços têm o mesmo peso e esta heurística não precisa ser executada.

3.1.5 Associação de serviços candidatos com papéis

Heurística: Agrupar serviços candidatos que estão associados a cada papel, listando papel e quantidade de serviços candidatos. Em seguida, definir pesos para cada agrupamento.

A associação de serviços candidatos com papéis permite identificar a relevância do serviço perante a quem o executa. Serviços candidatos executados apenas por sistemas devem ter um maior peso, dado que a funcionalidade será executada de forma

totalmente automatizada, sem qualquer intervenção humana e necessidade de desenvolvimento de interface gráfica. Já serviços que têm intervenção humana, implicam que deva existir uma interface implementada para permitir a interação da pessoa com o serviço. Estes serviços devem ter um peso mais baixo. Por outro lado, serviços executados tanto por sistema como também através de intervenção humana devem ter um peso intermediário entre os dois anteriores. A Tabela 4 apresenta proposta de pesos para os agrupamentos de serviços candidatos por papéis dos serviços identificados para o processo utilizado como estudo de caso neste trabalho.

Tabela 4 – Exemplo de agrupamento de papéis que utilizam (executam) o serviço

Papéis	Unidades de Negócio	Quantidade de serviços candidatos	Pesos
Apl1		13	5
Apl1; Papel1; Papel2	UN1; UN3;	1	2
Apl2		50	5
Apl2; Apl1; Papel1	UN1; UN4; UN3; UN2;	2	3
Papel1	UN1; UN3;	5	2
Papel2; Papel1	UN1; UN3;	22	2
Papel2; Papel1;	UN1; UN3; UN2; UN4;	1	3
Papel1 Apl2;	UN1; UN4; UN3; UN2;	3	3
Papel2; Papel1; Apl2;	UN3; UN1; UN2; UN3;	1	3
Papel1;	UN2; UN3;	8	2

Outro fator que pode ser considerado na atribuição dos pesos para serviços de acordo com papéis está na quantidade de recursos que podem ocupar o papel. Por exemplo, se o papel "Papel1" pode ser designado a 50 pessoas diferentes e o papel "Papel2" pode ser designado a 3 pessoas diferentes, isto pode significar que o uso do serviço associado à "Papel1" é muito maior do que o dos serviços associados a "Papel2". Logo, o peso para os agrupamentos considerando o papel "Papel1" deve ser maior. Por outro lado, se para a organização for muito mais importante automatizar via serviços aqueles que estão relacionados ao papel "Papel2", então maior importância deve ser dada para este agrupamento de serviços.

3.1.6 Cálculo da priorização de serviços candidatos

`Heurística: Calcular priorização de serviços candidatos de acordo de acordo com o somatório dos pesos definidos por cada heurística de priorização.`

Após definir os pesos para cada tipo de priorização, os valores devem ser atribuídos para cada serviço. Em seguida, o somatório dos pesos deve ser realizado. Os serviços com maior prioridade são aqueles com maior peso. A Tabela 5 apresenta uma amostra de serviços candidatos identificados a partir do processo utilizado para análise neste trabalho. A Tabela 6 apresenta o cálculo da prioridade para estes serviços.

Tabela 5 – Exemplos de serviços identificados

Id	Serviço candidato	Grau de reuso	Múlt. Inst.	Sistemas Impl.	Sistemas reutilizam	Papéis	UN	Req. Impl.	Req. Dem.
74	Serviço 74	1	NÃO	Apl1;	Apl1; Apl2	Apl1;	-	-	1
41	Serviço 41	2	NÃO	Apl2;	Apl2;	Apl2;	-	-	1
95	Serviço 95	2	NÃO	-	Apl2;	Papel 1; Papel 2;	UN1; UN3;	-	1
30	Serviço 30	2	NÃO	Apl2	Apl2;	Papel 1;	UN1; UN2; UN4;	-	1
109	Serviço 109	3	NÃO	-	Apl2;	Papel 2; Papel 1;	UN1; UN2; UN3; UN4;	-	1
13	Serviço 13	3	NÃO	Apl2;	Apl2;	Papel 1;	UN1; UN2; UN3; UN4	-	1
105	Serviço 105	4	NÃO	Apl2;	Apl2; Apl3	Papel 1; Apl 1;	UN1; UN2; UN3; UN4	-	1
101	Serviço 101	7	NÃO	Apl1;	Apl1; Apl2	Apl1; Papel 1; Papel 2;	UN1; UN2; UN3; UN4	-	1
98	Serviço 98	7	NÃO	Apl2;	Apl2; Apl3	Apl1; Papel 1;	UN1; UN2; UN3; UN4	-	1
92	Serviço 92	8	NÃO	Apl2; APL1;	Apl2; Apl1;	Apl2; APL1; Papel 1;	UN1; UN2; UN3; UN4	-	1

Tabela 6 – Exemplo de cálculo de priorização de serviços

Serviço	Peso						Prior.
	Múlt. Inst.	Reuso	Sist. Impl.	Sist. Reuso	Demanda	Papéis	
Serviço 74	0	0	2	4	1	5	11
Serviço 41	0	1	3	4	1	5	14
Serviço 95	0	1	3	4	1	5	14
Serviço 30	0	1	3	4	1	2	11
Serviço 109	0	2	1	4	1	2	10
Serviço 13	0	3	3	4	1	2	13
Serviço 105	0	3	3	6	1	3	16
Serviço 101	0	5	2	8	1	2	18
Serviço 98	0	5	3	6	1	3	18
Serviço 92	0	5	5	4	1	3	18

3.1.7 Detalhamento da priorização de serviços identificados a partir de fluxo

Heurística: Priorizar serviços de fluxo de acordo com as informações levantadas para caracterização dos mesmos.

A heurística de consolidação de serviços identificados a partir fluxo propõe o detalhamento das seguintes informações sobre estes serviços:

- Quantidade de atividades que compõem o fluxo:
  - Número de atividades automatizadas;
  - Número de atividades apoiadas por sistema.
- Modelos onde o fluxo aparece: listagem dos modelos onde o fluxo aparece, caso o fluxo apareça em mais de um modelo;

- Número de entidades do fluxo: número de entidades que são utilizadas nas atividades que compõem o fluxo, identificadas a partir dos modelos de dados associados aos clusters existentes nas atividades;
- Número de raias envolvidas no fluxo: número de raias diferentes onde existem atividades do fluxo;
- Números de sub-fluxos: número de sub-fluxos existentes no fluxo que gerou o serviço, por exemplo, em um fluxo de AND do qual o serviço foi identificado, quantos fluxos de OR, XOR e AND existem no sub-fluxo.

Para cada uma destas informações um peso deve ser atribuído, a fim de definir quais são os serviços mais prioritários dentre os serviços identificados a partir de fluxo. Uma proposta é fazer o agrupamento de cada uma destas informações e definir intervalos de valores para as quantidades encontradas. Por exemplo, sabendo que a quantidade de atividades que compõe o fluxo varia de 2 a 25, pode-se definir os seguintes pesos:

- Peso 0 para 2 atividades;
- Peso 1 para de 0 a 5 atividades;
- Peso 2 para de 6 a 10 atividades;
- Peso 3 para de 11 a 15 atividades;
- Peso 4 para de 16 a 20 atividades;
- Peso 5 para de 21 a 25 atividades.

Raciocínio análogo pode ser realizado para as demais informações. Em seguida procede-se com a atribuição dos pesos para cada serviço, e cálculo das prioridades. A Tabela 7 apresenta um exemplo de priorização de serviços candidatos identificados a partir de fluxos do processo utilizado neste trabalho.

Tabela 7 – Exemplo de priorização de serviços identificados a partir de fluxo

Serviço	Peso					Prioridade
	# atv. Autom.	# atv. Apoiadas	# modelos	# entidades	# sub-fluxos	
Serviço 110	4	0	1	-	-	5
Serviço 120	0	2	1	-	1	4

3.2 Granularidade inicial de serviços candidatos

Heurística: Elaborar matriz de granularidade de serviços candidatos e marcar serviços que têm maior ganho em disponibilizar como serviços físicos.

Após a priorização dos serviços candidatos, o próximo passo é analisar a granularidade dos serviços mais prioritários. Neste trabalho propomos a elaboração de uma matriz de granularidade semelhante à proposta por Marks e Bell [2006], a qual propõe que serviços mais finos (*finer-grained services*) devem ser disponibilizados na parte mais baixa, enquanto que serviços grossos (*coarse-grained services*) devem ser disponibilizados mais acima nesta matriz. A nossa proposta é considerar a dependência entre serviços para realizar esta análise, o que é semelhante à proposta de

Marks e Bell [2006]. Propomos que esta dependência seja identificada a partir do relacionamento entre elementos dos processos de negócio dos quais os serviços candidatos foram derivados. Por exemplo, requisitos de negócio geralmente referenciam regras de negócio. Logo, existe uma relação de dependência entre serviços candidatos identificados a partir de regras de negócio (mais finos) em relação aos serviços candidatos identificados a partir de requisitos de negócio (mais grossos). Outro exemplo, é que serviços identificados a partir de fluxos (serviços mais grossos) têm relação direta com os serviços candidatos identificados a partir dos elementos presentes em cada atividade (serviços mais finos). Por exemplo, serviços identificados a partir de regras de negócio, em geral, ficam posicionados na parte mais baixa da matriz, enquanto que serviços identificados a partir de requisitos de negócio ficam na parte mais acima e serviços identificados a partir de fluxo na parte mais acima ainda. Muitas destas dependências podem ser identificadas automaticamente, de acordo com a explicitação da relação entre os elementos dos modelos de processos dos quais os serviços foram identificados, por exemplo, relação entre regras de negócio e requisitos de negócio.

A Figura 4 apresenta um exemplo de matriz de granularidade para alguns dos serviços candidatos identificados a partir do modelo de processos utilizado neste trabalho. Como o grau de reuso destes serviços é muito baixo, uma decisão que poderia ser tomada é disponibilizar como serviço físico apenas o serviço Serviço 1 (o qual foi marcado com *físico*) ficando os demais serviços candidatos encapsulados dentro deste serviço.

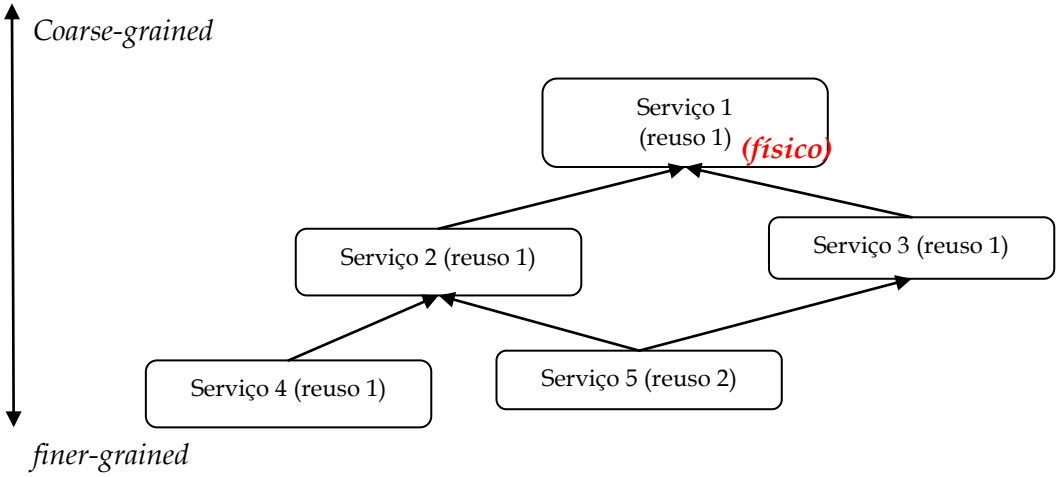


Figura 4 – Grafo de granularidade

Propomos também que serviços mais importantes (de acordo com a priorização calculada) sejam colocados mais ao topo na matriz, indicando que estes serviços candidatos devem ser considerados para serem disponibilizados como serviços físicos. Um exemplo é apresentado na Figura 5, no qual o serviço candidato Serviço 4 foi considerado com prioridade alta. Neste caso, os serviços candidatos Serviço 1 e Serviço 4 foram marcados como propostas de serviços físicos. Logo, o resultado da granularidade inicial de serviços candidatos pode levar a descartar serviços candidatos de níveis mais baixos. Estes então serão implementados como componentes encapsulados em serviços ou como componentes compartilhados por serviços, mas nunca expostos diretamente no barramento de serviços.

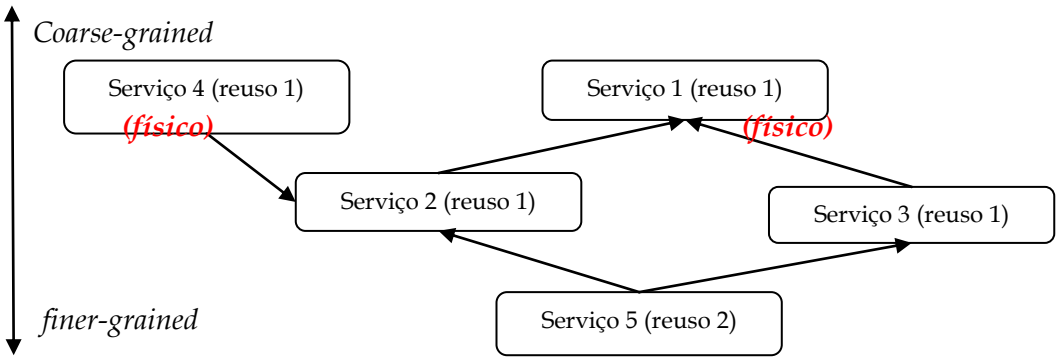


Figura 5 – Exemplo de grafo de granularidade considerando a importância de serviços

3.3 Agrupamento de serviços de dados

Heurística: Associar serviços candidatos de dados com grupos de entidades indicando as operações CRUD (*Create*, *Retrieve*, *Update* e *Delete*) que o serviço realiza sobre entidades do grupo. Em seguida, agrupar serviços de acordo com operações.

Heurística: Definir canais de comunicação entre grupos de serviços candidatos de acordo com operações *Read*.

Heurística: Projetar serviços buscando reduzir o custo de comunicação entre eles.

Serviços de dados (também chamados de serviços de entidade) são muito genéricos e reutilizáveis por natureza, dado que encapsulam o acesso a dados que pode ser realizado por qualquer aplicação que deseja ler, criar, atualizar ou remover um dado da base de dados. Erl [2005] aponta que estes serviços não estão fortemente acoplados a processos de negócio, dado que proveem uma interface que é mais orientada aos dados do que aos processos. Dessa forma, a análise destes serviços deve ser feita a partir das entidades que eles operam. Estas entidades podem ser identificadas diretamente a partir do modelo de processos de negócio, por exemplo, Azevedo *et al.* [2008b] propõe listar as entidades associadas aos modelos de dados ligados aos clusters para todos os serviços candidatos de dados identificados a partir de clusters. Caso esta informação não esteja disponível, então é necessário analisar modelos Entidade-Relacionamento, diagramas de classe ou realizar levantamentos com administradores de dados e usuários dos serviços, a fim de levantar as entidades manipuladas pelos serviços.

Tendo em mãos estas entidades, propomos listar em uma tabela os serviços candidatos relacionando-os às entidades e atributos das entidades que eles manipulam. Em seguida, deve ser realizado um agrupamento dos serviços candidatos de acordo com o grau de semelhança considerando as informações manipuladas. Após a explicitação do agrupamento, serviços candidatos muito semelhantes (que tratam a mesma entidade ou grupo de entidades) devem ser generalizados em um único serviço candidato ou serem agrupados em um mesmo serviço físico.

A Tabela 8 lista a associação de serviços candidatos identificados a partir de clusters em relação ao processo estudado neste trabalho.

Tabela 8 – Exemplo de associação de serviços candidatos com grupos de entidades

Serviço	Grupos de Entidades						
	Entidade 1	Entidade 2	Entidade 3	Entidade 4	Entidade 5	Entidade 6	Entidade 7
Serviço 1	R						
Serviço 2		C					
Serviço 3	R						
Serviço 4			C				
Serviço 5			C				
Serviço 6				R			
Serviço 7		R		C			
Serviço 8					R		
Serviço 9		R			C		
Serviço 10		R				C	
Serviço 11		R				C	
Serviço 12					R		
Serviço 13				R			
Serviço 14	C				R		
Serviço 15							R
Serviço 16		R					R
Serviço 17		C					R
Serviço 18					R		

A Tabela 9 apresenta os agrupamentos dos serviços candidatos de acordo com as manipulações dos grupos de entidades. Como resultado desta análise pode-se definir que serviços candidatos de um mesmo grupo devem ser agrupados em um único serviço físico, o qual é responsável por disponibilizar um método web correspondente a cada serviço candidato. Esta forma de agrupamento tem como base a proposta de Jamshid *et al.* [2008]. Portanto, são agrupados em um mesmo grupo os serviços candidatos que realizam operações *Create* e *Update* no mesmo grupo de entidades. Linhas que têm apenas uma operação *Read* são agrupadas no grupo mais próximo, considerando operações *Create* e *Update* como, por exemplo, as linhas 2 e 3 da Tabela 9. Por outro lado, quando não existir coluna com *Create* e *Update* para o grupo de entidades, então considera-se o significado do serviço candidato como, por exemplo, a última e penúltima linhas da Tabela 9.

No caso em que um grupo de serviços candidatos que manipula um grupo de entidades realizar uma operação *Read* sobre um outro grupo de entidades manipulado por outro grupo de serviços candidatos, temos um canal de comunicação entre estes dois grupos de serviços candidatos. Na Tabela 9, isto é exemplificado pelo canal de comunicação entre o grupo de serviços G1 e o grupo de serviços G4.

Durante a fase de projeto, é interessante projetar serviços buscando, sempre que possível, que aqueles serviços que se comunicam (possuem a marcação H - Tabela 9) sejam dispostos no mesmo pacote. O objetivo é diminuir o custo de comunicação entre estes serviços. Contudo, a disposição dos serviços em um mesmo pacote pode aumentar o acoplamento entre os serviços, dependendo da forma com que a comunicação entre os serviços seja codificada. Quando a comunicação é realizada ponto-a-ponto, a disposição em um mesmo pacote diminui o tráfego de rede entre os serviços e até mesmo diminuir o tempo de conversão das mensagens XML, caso o *framework* adotado não trafegue mensagens XML entre serviços contidos no mesmo pacote. Por outro lado, ao utilizar uma abordagem com mediação de mensagens para evitar o acoplamento, a disposição em um mesmo pacote poderá não apresentar melhorias de desempenho.

Tabela 9 – Agrupamento de serviços candidatos de acordo com grupos de entidades manipuladas

Serviço		Grupos de Entidades						
		Entidade 1	Entidade 2	Entidade 3	Entidade 4	Entidade 5	Entidade 6	Entidade 7
G1	Serviço 14	C				H		
	Serviço 1	R						
	Serviço 3	R						
G2	Serviço 18		C					H
	Serviço 2		C					
G3	Serviço 7		H		C			
	Serviço 6				R			
	Serviço 13				R			
G4	Serviço 9		H			C		
	Serviço 8					R		
	Serviço 12					R		
	Serviço 17					R		
G5	Serviço 10		H				C	
	Serviço 11		H				C	
G5	Serviço 4			C				
	Serviço 5			C				
G6	Serviço 16		H					R
	Serviço 15							R

3.4 Hierarquia de serviços de dados

Heurística: Serviço de dados deve encapsular acesso a diferentes bases de dados, bem como transformações necessárias de cada banco de dados para o modelo canônico.

Se o conceito armazenado nas diferentes bases de dados for acessado de forma integrada, não havendo reuso no acesso às informações das bases de dados de forma independente, então deve ser projetado um único serviço de dados. Caso contrário deve ser projetada uma hierarquia de serviços: serviços de acesso a dados (para as diferentes bases de dados) e serviço de integração (para integrar os dados retornados pelos serviços de acesso a dados).

Byrne *et al.* [2008] e Py *et al.* [2009] propõem uma arquitetura de serviços englobando diferentes tipos de serviços, por exemplo:

- Serviço de acesso a dados - que tem a responsabilidade de manipular dados (executar operações CRUD – *Create*, *Retrieve*, *Update* e *Delete*) de uma determinada base de dados;
- Serviço de integração – que tem a responsabilidade de invocar os serviços de acesso a dados para obter dados de diferentes bases de dados e de realizar a integração destes dados segundo metadados que descrevem como os dados devem ser integrados.

O serviço de integração é um serviço composto que invoca serviços de acesso a dados. Logo, uma conseqüência desta arquitetura é a necessidade de realizar

chamadas para diferentes serviços, aumentando o custo de comunicação da transação e a necessidade da transformação das estruturas de dados recebidas dos serviços de acesso aos dados para a estrutura de dados a ser retornada pelo serviço de integração. Por outro lado, ao seguir esta abordagem, são definidas camadas de serviços que podem ser reutilizados de forma diferente.

3.5 Agrupamento de serviços de lógica

Heurística: Agrupar serviços de lógica de acordo com padrões de uso na organização.

O agrupamento de serviços de lógica deve levar em consideração o contexto de uso da lógica representada pelo serviço. Uma proposta de passo-a-passo para execução desta heurística é apresentada a seguir:

1. Identificar conceito mais abstrato utilizado pela organização em diferentes áreas;
2. Identificar os serviços candidatos de lógica que tratam o contexto deste conceito (e conceitos relacionados), considerando as visões das diferentes áreas da organização;
3. Generalizar serviços candidatos que realizam operações muito semelhantes sobre o conceito.
4. Agrupar em um mesmo pacote (ou em um mesmo serviço), serviços candidatos de lógica que operam sobre o contexto do conceito.

3.6 Especificação de serviços

3.6.1 Modelo dos tipos de dados utilizados pelo serviço

Heurística: Modelar os tipos de dados utilizados pelo serviço em um diagrama UML. Se um modelo canônico estiver sendo utilizado, explicitar a relação entre as classes e atributos do tipo de dados e as classes do modelo canônico.

Os modelos dos tipos de dados utilizados pelo serviço devem ser elaborados a fim de evitar a replicação dos mesmos tipos de dados em projetos diferentes, aumentar o reuso de tipos e documentar os tipos de dados existentes.

Caso seja uma diretriz da organização a elaboração de um modelo canônico, então, é importante que os modelos dos tipos de dados utilizados pelo serviço referenciem o modelo canônico indicando quais classes e atributos são utilizados. A referência entre modelos pode ser feita através da explicitação de dependências entre classes e atributos ou através de marcações no modelo canônico.

3.6.2 UML *Profiles*

Heurística: Documentar a especificação dos serviços utilizando um *profile* UML para SOA.

As decisões tomadas durante a fase de análise de serviços devem ser especificadas utilizando uma linguagem de modelagem de artefatos de *software*, como a UML.

Existem diversas ferramentas que apóiam a modelagem utilizando UML. A especificação UML permite criar extensões da linguagem chamados de *profiles* [OMG, 2009a]. Estes *profiles* UML permitem adicionar estereótipos na linguagem para representar artefatos mais específicos. Para modelagem de serviços, pode-se utilizar de *profiles* próprios para especificação de serviços web, por exemplo. Numa abordagem SOA, contudo, é importante que a especificação dos serviços se dê através de múltiplas visões de análise, como, por exemplo, a especificação técnica do serviço, a especificação do ponto de vista da arquitetura de *software* da empresa, o relacionamento do provedor do serviço com seus consumidores, etc.

Na literatura, alguns *profiles* UML foram propostos tanto no meio acadêmico [Heckel *et al.,* 2003; López-Sanz *et al.,* 2007], quanto empresarial [Johnston, 2005a; OMG, 2009b]. Dentre estas propostas, podemos destacar os *profiles* que estão atualmente disponíveis para utilização em ferramentas *case*, como a proposta de [Johnston, 2005a], criada pela IBM e disponibilizada na suíte da Rational, e a proposta conjunta da OMG [2009b], que, embora ainda esteja em fase de refinamento da especificação, pode se tornar um padrão para especificação de artefatos em SOA. Este *profile* está disponível para avaliação na ferramenta ModelPro [ModelDriven, 2009].

A

Tabela **10** apresenta um comparativo dos estereótipos que podem ser utilizados nos *profiles* UML da IBM e da OMG.

Tabela 10 – Estereótipos disponíveis nos UML *profiles* da IBM e OMG

Finalidade do estereótipo	IBM [Johnston, 2005a]	SoaML (OMG, 2009b]
Descrição do serviço	Service ServicePartition ServiceSpecification ServiceGateway	Service ServiceCapability ServiceContract ServiceInterface Property
Troca de mensagens	Message MessageAttachment	MessageType Attachment Request
Processo	ServiceProvider ServiceConsumer ServiceCollaboration	CollaborationUse Participant Agent Milestone ServicesArchitecture
Conectores	ServiceChannel	ServiceChannel ConnectableElement
Integração com a classificação RAS (*Reusable Asset Specification*) da OMG		Catalog Categorization Category CategoryValue DescriptorGroup
Integração com a especificação BMM (*Business Motivation Model*) da OMG		MotivationElement MotivationRealization
Outros		ValueObject

3.6.3 SoaML

Esta seção apresenta alguns exemplos de uso de alguns elementos principais da especificação SoaML.

Assim como uma interface UML, uma ServiceInterface pode ser o tipo de uma porta do serviço. A interface do serviço tem como característica adicional permitir especificar um serviço bi-direcional – onde tanto o provedor quanto o consumidor têm responsabilidades no envio e recebimento de mensagens e eventos. A interface de serviço é definida da perspectiva do provedor de serviço usando três elementos primários: as interfaces necessárias, a classe ServiceInterface e o protocolo Behavior. A Figura 6 apresenta um exemplo de especificação de interface de serviço utilizando o elemento ServiceInterface.

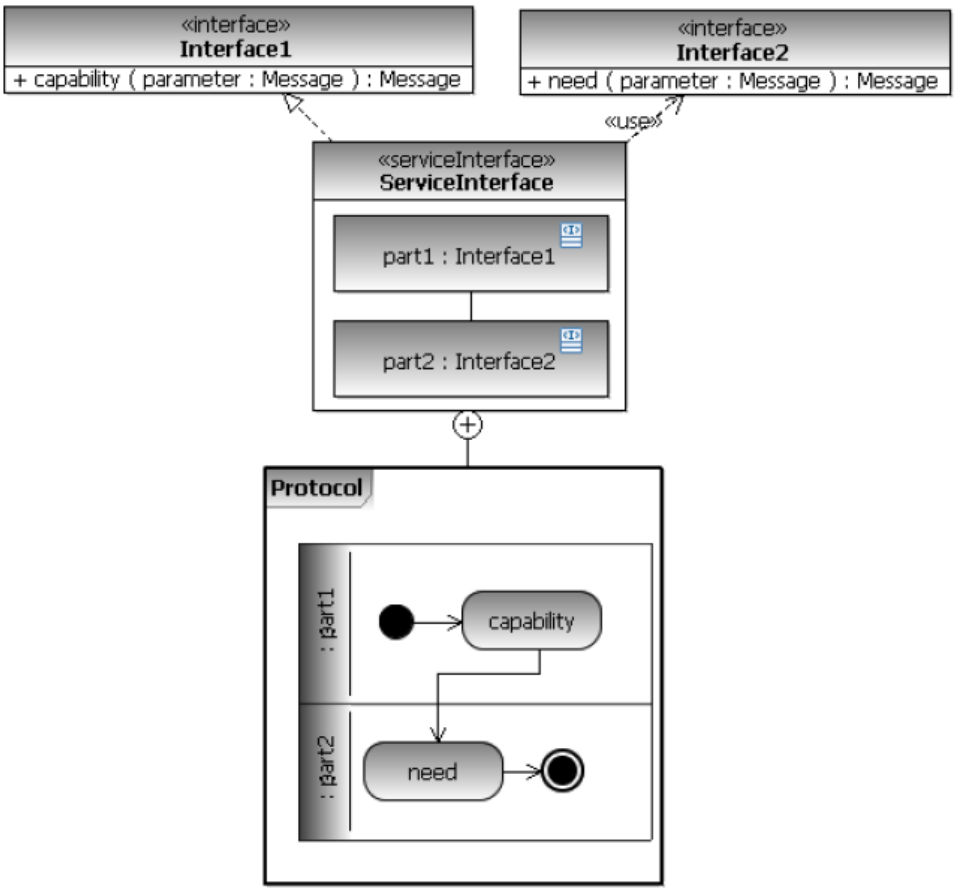


Figura 6 – Exemplo de especificação de interface de serviço

No exemplo acima, está representada uma interface genérica de serviço. Interface1 representa uma interface provida com suas capacidades (*capability*) (representa uma funcionalidade que a interface fornece, um método), enquanto Interface2 representa uma interface que define o que é esperado que os consumidores façam em resposta às requisições dos serviços. A *ServiceInterface* tem duas partes, part1 e part2, as quais representam os pontos de acesso (*endpoints*) entre as requisições do consumidor e os serviços providos. Part1 é do tipo Interface1, indicando que representa o provedor ou o lado da conexão do serviço. Parte 2 é do tipo Interface2, indicando que representa o consumidor ou o lado de requisição da conexão.

Uma das principais características da SoaML é a capacidade de representar um contexto de interação de serviços e participantes e representar o relacionamento entre esses contextos de interação. Dessa forma, é possível visualizar, num alto nível, a arquitetura de serviços da organização. Para tal, o estereótipo *ServicesArchitecture* é disponibilizado. Uma *ServicesArchitecture* é uma rede de papéis de participantes provendo e consumindo serviços para realizar um propósito. A arquitetura de serviços define os requisitos dos tipos de participantes e realizações dos serviços que preenchem esses papéis. Com a definição de papéis, a SoaML permite modelar pessoas, organizações e sistemas que colaboram, sem se preocupar, em um primeiro momento, em quais são esses sistemas, pessoas ou organizações.

A arquitetura de serviços é modelada em dois níveis de granularidade. O primeiro nível corresponde à arquitetura de serviços da comunidade, uma visão de alto nível apresentando como participantes independentes trabalham em conjunto para realizar

um propósito. O segundo nível corresponde à arquitetura de serviços do participante e especifica a arquitetura para um participante em particular.

A arquitetura de serviços da comunidade é definida usando um diagrama UML de colaboração (Figura 7).

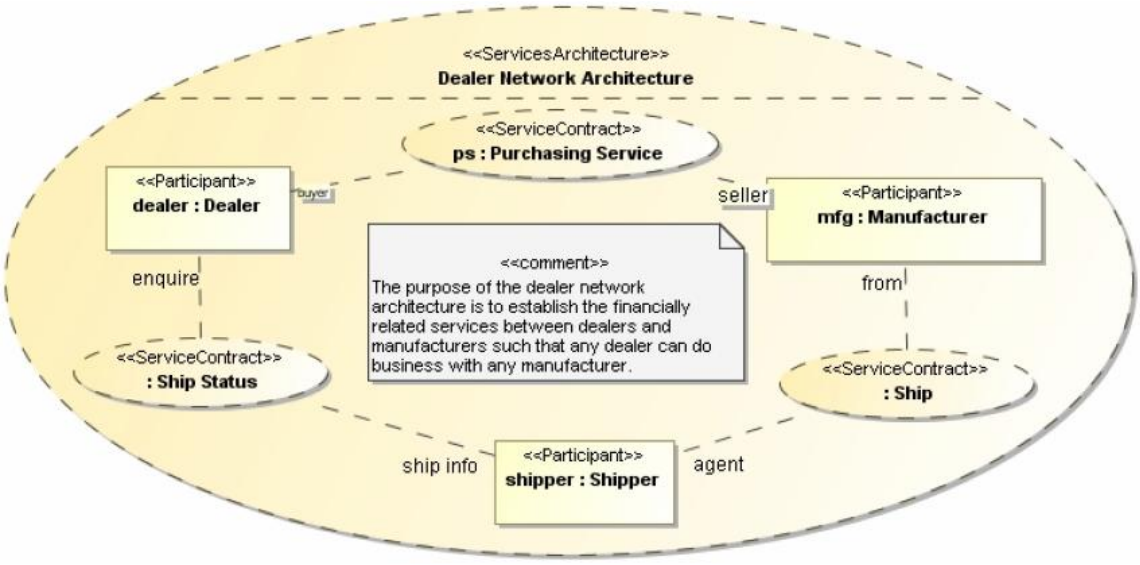


Figura 7 – Arquitetura de serviços da comunidade

O propósito da colaboração acima é ilustrar como tipos de entidades trabalham em conjunto para algum propósito. Colaborações são baseadas no conceito de papéis para definir como entidades estão envolvidas na colaboração (como e porquê elas colaboram) sem depender de que tipo de entidade está envolvida (por exemplo, pessoa, organização ou sistema). No exemplo, a arquitetura Dealer Network Archicture define um ambiente onde 3 participantes (um fabricante – *manufacturer* –, um revendedor – *dealer* –, e uma empresa de envio – *shipper*–) se relacionam através dos serviços de compra (*Purchasing Service*), envio (*Ship*) e estado da remessa (*Ship Status*). Assim, um revendedor compra um item de um fabricante através do serviço de compra. O fabricante repassa o item para uma empresa de envio através do serviço de envio e o revendedor, por sua vez, pode verificar a condição do envio do item que foi enviado pela empresa de envio através do serviço de verificação do estado da remessa (*Ship Status*).

A arquitetura de serviços do participante, por sua vez, ilustra como sub-participantes e colaboradores externos trabalham em conjunto e quase sempre representa um processo de negócio. Um *ServicesArchitecture* (tanto da colaboração quanto do participante) pode ser compostos de outras arquiteturas de serviços e contratos de serviços. Como mostra a Figura 8, participantes (*Participant*) são classificadores definidos tanto pelos papéis que executam na arquitetura de serviços (o papel do participante) quanto nos requisitos do "contrato" das entidades que executam esses papéis. Cada tipo de participante pode "executar um papel" em um número de arquitetura. Os requisitos são satisfeitos quando o participante tem portas de serviço (Service ports) que tem um tipo compatível com os serviços que eles precisam prover ou consumir.

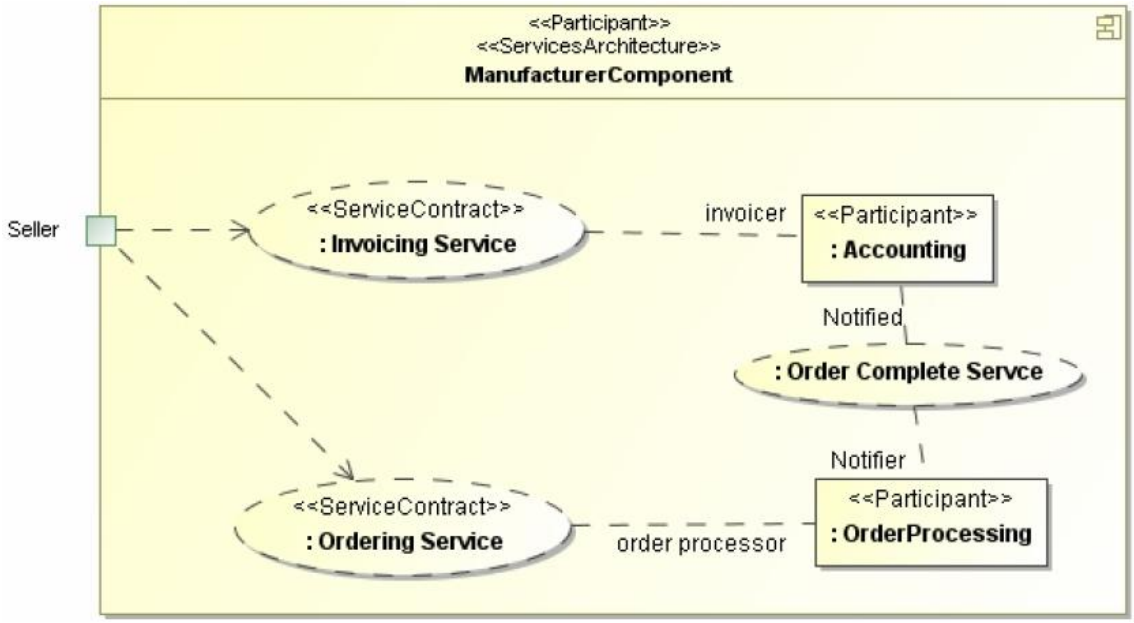


Figura 8 – Arquitetura de serviços do participante

A parte chave de um serviço é o seu contrato (*ServiceContract*). Um *ServiceContract* define os termos, condições, interfaces e coreografia que os participantes precisam estar de acordo para que (direta ou indiretamente) o serviço possa ser usado. Um contrato de serviço é ligado tanto nos provedores quanto nos consumidores daquele serviço. A base para descrição de um contrato de serviço também é um diagrama de colaboração da UML que foca nas interações envolvidas em prover um serviço, como mostra a Figura 9.

Figura 9 – Contrato de um serviço

Uma parte importante do *ServiceContract* é a coreografia. A coreografia é a especificação do que é transmitido e quando isto é transmitido entre parceiros. A coreografia especifica trocas entre os parceiros – os dados, artefatos, obrigações entre as partes. Ela especifica o que acontece entre os participantes provedores e consumidores sem definir seus processos internos – seu processo interno tem que ser compatível com seu contrato de serviço (*ServiceContract*).

Uma coreografia de um *ServiceContract* é um comportamento UML que pode ser modelado como um diagrama de interação ou um diagrama de atividade, como mostra a Figura 10.

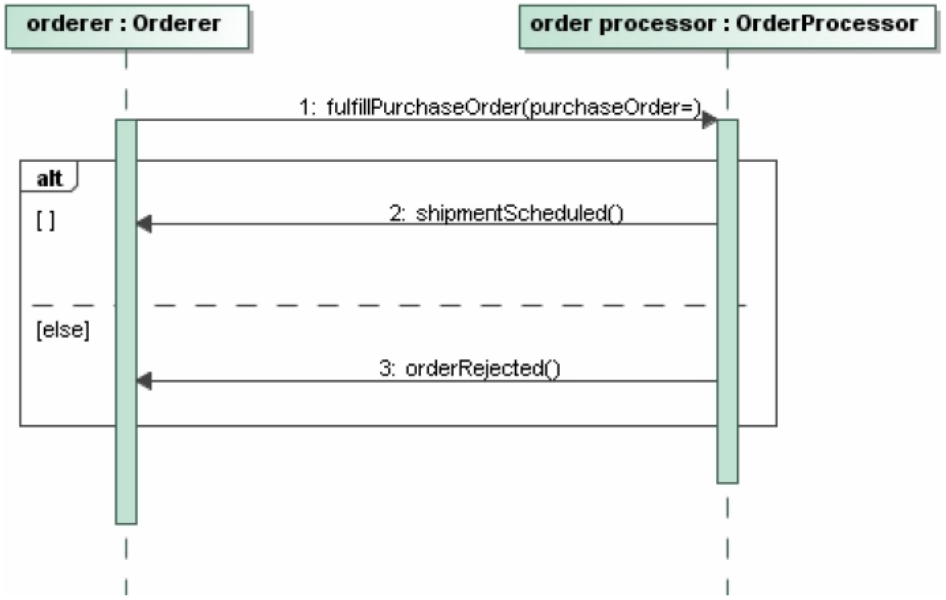


Figura 10 – Coreografia de serviços

3.6.4 UML *profile* da IBM

Simon Johnston [2005a] apresenta um *profile* UML criado pela IBM para descrição de serviços nas ferramentas Rational. O *profile* UML, para descrição de serviços, criado pela IBM Rational é utilizado para descrever atividades de desenvolvimento do ciclo de vida do serviço e visões para diferentes *stakeholders*. A Figura 11 apresenta o metamodelo do *profile* criado pela IBM utilizando os estereótipos definidos para representar elementos de descrição de serviços. A figura demonstra os detalhes do *profile*, apresentando cada estereótipo com sua metaclasse usando a notação de extensão (setas com ponta preenchida). As restrições estão descritas nesse modelo como notas.

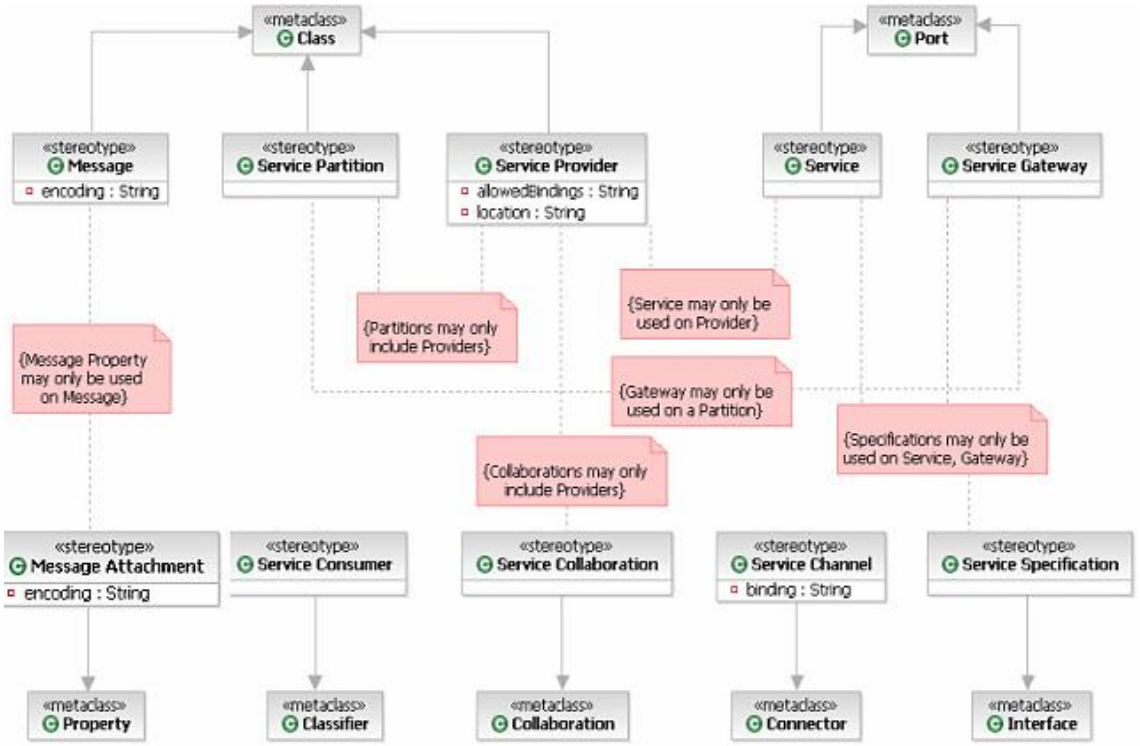


Figura 11 – *Profile* da IBM para descrição de serviços

Um serviço (Service) e um *gateway* (Service Gateway) são especializações da metaclasse Port e são especificados através de uma especificação de serviço (Service Specification). Serviços são usados por provedores de serviços (Service Provider), os quais participam de colaborações com outros serviços (Service Colaboration). Provedores de serviços podem participar de uma partição (Service Partition) de algum *gateway* (Service Gateway). Serviços podem enviar mensagens (Message), as quais podem conter anexos (Message Attachment).

4 Proposta de metodologia de projeto de serviços

Esta seção apresenta heurísticas e boas práticas para projeto de serviços.

4.1 Heurísticas para projeto de serviços

Esta seção apresenta heurísticas para projeto de serviços que devem ser considerados pelo projetista ao desenvolver serviços para uma iniciativa SOA. Estas heurísticas são baseadas nos princípios para serviços em SOA, extraídos de [Josuttis, 2007; Portier, 2007; Hewitt, 2009]. Vale notar que alguns princípios dessa lista são originários de outros paradigmas, como orientação a objetos.

4.1.1 Auto-contido

Heurística: Priorizar implementação de serviços auto-contidos. Caso seja necessário que um serviço invoque outro serviço, considerar o uso de uma tecnologia de orquestração de serviços.

Segundo Josuttis [2007], todas as definições de SOA consideram que existe um objetivo durante o projeto de serviços de fazê-los auto-contidos, ou seja, independentes, autônomos, autárquicos. Contudo, algumas dependências sempre existem. Por exemplo, serviços podem compartilhar ao menos um tipo fundamental como string. Apesar disso, deve-se ter cuidado com o compartilhamento de tipos de dados mais complexos. Já Hewitt [2009] apontam que deve-se tomar cuidado em um serviço invocar diretamente outro serviço (ou seja, o serviço ser ele mesmo um cliente de outro serviço), isto pode reduzir significativamente o contexto do negócio disponível para reuso do serviço. Se esta situação ocorrer, considerar o uso de uma tecnologia de orquestração como, por exemplo, BPEL. Assim como um serviço encapsula um sistema ou um subsistema, uma orquestração encapsula um conjunto de serviços que, em conjunto, formam um fluxo. Mas uma orquestração é exposta ela mesma como um serviço, ou seja, ela se apresenta para o cliente como se fosse outro serviço.

O objetivo, enfim, é minimizar as dependências para que a abordagem SOA seja mesmo apropriada para sistemas distribuídos com diferentes proprietários. Quanto menos dependências, modificações ou erros em um sistema, menos conseqüências serão produzidas em outro sistema. Esse princípio está conceitualmente muito próximo do princípio de baixo acoplamento, descrito na seção seguinte, logo, este assunto será mais detalhado na próxima seção.

4.1.2 Baixo acoplamento

Acoplamento significa quantas entidades ou sistemas dependem uma das outras. No contexto de SOA, pode-se pensar em acoplamento entre a especificação de um serviço e seu provedor ou sua implementação, ou acoplamento entre um provedor de serviços e um consumidor. Para dar suporte à agilidade prometida em SOA, serviços devem ter baixo acoplamento.

Um provedor não deve ter conhecimento sobre uma instância específica de um consumidor e vice-versa. Trocar um provedor ou a implementação de um serviço deve ter efeitos mínimos nos consumidores. A Tabela 11 lista formas possíveis de baixo acoplamento em SOA.

Tabela 11 – Formas possíveis de baixo acoplamento em SOA, adaptado de [Krafzig *et al.*, 2004]

	ALTO ACOPLAMENTO	BAIXO ACOPLAMENTO
Conexões físicas	Ponto a ponto	Via mediador
Tipo de comunicação	Síncrono	Assíncrono
Modelo de dado	Tipos complexos comuns	Somente tipos simples comuns
Sistema de tipo	Forte	Fraco
Padrão de interação	Navegar através de árvores de tipos complexos	*Data-centric,* mensagem alto-contida
Controle da lógica do processo	Controle central	Controle distribuído
Ligação (*binding*)	De forma estática	Dinamicamente
Plataforma	Forte dependência de plataforma	Independente de plataforma

	ALTO ACOPLAMENTO	BAIXO ACOPLAMENTO
Controle de transações	2PC (Two-phase commit)	Compensação
Implantação	Simultânea	Em momentos diferentes
Versionamento	Atualizações explícitas	Atualizações implícitas

É razoável notar, contudo, que:

(1) a tabela acima não é uma lista de características técnicas que um sistema precisa implementar para que o ambiente seja considerado uma abordagem SOA, ou seja, alguns pontos podem ser flexibilizados para atender uma melhor solução;

(2) o baixo acoplamento possui um custo. O sistema pode tornar-se mais complexo, resultando em um maior esforço de desenvolvimento para sua manutenção evolutiva. Como exemplos, podemos citar: o uso de comunicação assíncrona (seção 4.1.2.1) tentativa de harmonização dos tipos de dados (seção 4.1.2.2); uso de mediadores (seção 4.1.2.3); verificação de tipos de dados (seção 4.1.2.4); dentre outras questões.

4.1.2.1 Comunicação assíncrona

Heurística: Para cada serviço, priorizar implementação utilizando comunicação assíncrona. Se não houver restrição de desempenho ou a utilização do padrão assíncrono levar a uma implementação muito complexa, então utilizar padrão síncrono, mas justificando.

Um dos exemplos mais conhecidos de baixo acoplamento é a comunicação assíncrona. Esse tipo de comunicação geralmente compreende um remetente que envia uma mensagem para um destinatário, o qual retornará a resposta para a mensagem. Porém, ao contrário da comunicação síncrona, o remetente, após enviar a mensagem, continuará seu trabalho enquanto a mensagem de resposta não for recebida. Neste caso, o destinatário não precisa estar *online* no momento do envio da mensagem. Quando o destinatário se torna disponível, a mensagem é entregue a ele para ser processada.

Um problema com comunicação assíncrona ocorre quando o remetente necessita de uma resposta para a mensagem enviada. Como a mensagem de resposta pode demorar a chegar e o remetente continua seu trabalho após o envio da mensagem, é preciso identificar as mensagens para que o remetente identifique a resposta corretamente. Uma possível solução é associar um ID à mensagem (também chamado de "correlation ID"). Uma vez com a resposta em mãos, o serviço precisa correlacionar a resposta com a mensagem enviada, o que requer conhecimento de algum estado inicial e contexto de quando a mensagem foi enviada. Esta tarefa demanda algum esforço. A situação piora quando um número muito grande de mensagens é enviado assincronamente e é necessário processar essas mensagens de acordo com a ordem de envio. Para ajudar em tal cenário, uma ferramenta que dê suporte ao padrão *WS-ReliableMessaging*[1] pode ser muito útil e diminui a complexidade do serviço para tratamento do recebimento de mensagens.

[1] WS-ReliableMessaging descreve um protocolo que permite que mensagens sejam entregues de forma confiável entre aplicações distribuídas a presença de falhas de componentes de software, sistema ou de rede. (http://www.ibm.com/developerworks/library/specification/ws-rm/)

4.1.2.2 Tipos de dados heterogêneos

Heurística: Antes de criar um tipo de dado na implementação do serviço, verificar os tipos de dados existentes no modelo canônico, caso exista uma diretriz organizacional para sua definição.

Heurística: Não definir tipos de dados homogêneos para todos os contratos dos serviços.

A harmonização dos tipos de dados é um grande problema em ambientes distribuídos. Não há dúvida que tudo se torna muito mais fácil se os tipos de dados estão compartilhados em todos os sistemas. Segundo Josuttis [2007], quando a orientação a objetos se tornou famosa, ter um modelo de objeto de negócio comum se tornou um objetivo geral. Entretanto, essa abordagem se tornou um problema quando aplicada a grandes sistemas. A primeira razão é organizacional: é muito difícil entrar em acordo sobre tipos de dados comuns. Existem visões e interesses que variam muito de sistema pra sistema, uma vez que os diversos sistemas possuem diferentes proprietários. Cedo ou tarde o preço da harmonização se torna muito alto, pois é muito caro manter todos os sistemas sincronizados e, quando um novo serviço for adicionado, a heterogeneidade será introduzida novamente. Se tipos de dados não estão harmonizados, então é preciso criar mapeamentos entre eles.

Apesar da utilização de tipos de dados heterogêneos diminuir o acoplamento, a existência de uma diretriz organizacional para definir um modelo canônico de dados pode ser útil para manter a reusabilidade do código implementado. Para a definição do contrato dos serviços, contudo, a homogenização dos tipos de dados dificulta a criação de visão sobre os tipos de dados e pode aumentar desnecessariamente o tamanho das mensagens entre os serviços.

4.1.2.3 Mediadores

Heurística: Utilizar mediadores para invocação de serviços.

Essa forma de baixo acoplamento tem a ver com a forma com que uma chamada de serviço é realizada. Na abordagem ponto-a-ponto o remetente envia a requisição para um sistema específico usando um endereço físico. É o equivalente a enviar uma carta para o endereço da casa de uma dada pessoa. Essa é a abordagem altamente acoplada. O que acontece se a pessoa muda de casa? O que acontece se o serviço está fora do ar ou está sobrecarregado de mensagens? Mecanismos para falhas e balanceamento de carga são necessários.

Em princípio, existem dois tipos de mediadores: (1) o primeiro tipo informa o ponto de acesso correto para o serviço, antes de o consumidor enviar a resposta. Ou seja, a conexão continua sendo ponto a ponto, mas com a ajuda de mediadores o consumidor envia a mensagem para diferentes endereços. O consumidor pede pelo serviço usando um nome simbólico e o mediador envia o endereço físico do serviço; (2) o segundo tipo escolhe o ponto de acesso correto para a requisição depois que o consumidor envia a mensagem. Neste caso, o consumidor envia a requisição para um nome simbólico e a infraestrutura (rede, *middleware*, ESB) roteia o pedido para o sistema correto de acordo com as regras de roteamento inteligente.

4.1.2.4 Verificação fraca de tipos

`Heurística: Utilizar verificação de tipos nos mediadores quando contrato dos serviços estiver estável.`

Outro exemplo da complexidade do baixo acoplamento tem a ver com o problema de verificar ou não tipos de dados e quando fazê-lo. É sabido que quanto mais cedo um erro for descoberto, melhor. Por essa razão, linguagens de programação com verificação de tipos parecem ser melhores do que as que não fazem essa verificação, uma vez que elas detectam possíveis erros em tempo de compilação. Para que uma infraestrutura SOA (como um ESB) verifique tipos, é necessária alguma informação sobre esses tipos. Se os tipos de dados estão descritos em XML, então o ESB necessita do *XMLSchema* correspondente. Como conseqüência, qualquer modificação no tipo não vai afetar somente o consumidor e o provedor, mas também o ESB. Por essa razão, pode ser uma boa idéia manter o ESB genérico. Em geral, a mudança nos tipos deveriam apenas afetar quem usa o tipo como contrato, não quem transfere os dados.

É preciso perceber, contudo, que os tipos de dados são processados pelos sistemas. Então, qualquer inconformidade no tipo de dados recebido por um sistema pode resultar em um processamento errôneo. Assim, é razoável realizar uma verificação de tipos nestes sistemas. O problema principal é quando realizar essa verificação. A grande questão aqui é quão estável o dado está. Digamos que o sistema está sendo utilizado por muito pouco tempo em um ambiente distribuído, então é possível que muitas alterações no formato do tipo de dado ainda poderão ser necessárias por causa de requisitos novos que foram surgindo. Se, por outro lado, o tipo de dado é considerado estável, a verificação de tipos poderia ser aconselhável.

4.1.2.5 Padrão de interação

`Heurística: Utilizar tipos de dados comuns em diversas linguagens de programação.`

Uma forma de dependência de plataforma tem a ver com quais padrões de interação são usados na assinatura dos serviços, por exemplo: quais paradigmas de programação são fornecidos para projetar as interfaces do serviço; quais são os tipos fundamentais de dados e como eles podem ser combinados. Várias questões precisam ser consideradas:

- Somente caracteres são possíveis de serem usados ou outros tipos de dados fundamentais também são possíveis (inteiros, reais, etc)?
- Pode-se utilizar enumerações (conjuntos limitados de caracteres)?
- É possível restringir valores possíveis (por exemplo, definir formatos de caracteres)?
- Pode-se construir tipos mais complexos (estruturas, etc)?
- Pode-se construir tipos seqüenciais (arrays, listas)?
- Pode-se projetar relações entre os tipos (herança, extensões, polimorfismo)?

Quanto mais complexos forem os construtores de programação, de forma mais abstrata você pode programar. Contudo, você também terá mais problemas em mapear dados para plataformas que não suportem de forma nativa um certo construto.

Josuttis [2007] recomenda ter um conjunto básico de tipos fundamentais que possam ser compostos com outros tipos (estruturas, *records*, etc) e sequências (*arrays*). É necessário tomar cuidado com enumerações, herança e polimorfismo (mesmo quando XML suportar isto). Em resumo, segundo o autor, deve-se ser conservador com os tipos de dados.

4.1.2.6 Controle de transações

`Heurística: Definir operações de compensação para as operações dos serviços.`

Controle de transações pressupõe um maior acoplamento entre o consumidor e o provedor do serviço. Em alguns casos, o controle de transação pode ser flexibilizado e ser adotado um controle de compensações. A vantagem da compensação é que a atualização de dados dos sistemas não precisa ser feita de forma síncrona. A linguagem BPEL suporta compensações. Contudo, ao adotar controle por compensações, deve-se explicitamente prover e chamar serviços para reverter erros anteriores.

Compensações é uma forma interessante de baixo acoplamento. A abordagem comum para resolver problemas de transação é criar um contexto de transação comum usando técnicas como *two-phase commit* (2PC). Com essa abordagem, primeiro realiza-se todas as modificações em todos os serviços que lidam com o dado, exceto o aceite de atualização final do dado. Então, se nenhum serviço sinalizar um problema, então o serviço controlador realiza a atualização em todos os sistemas. 2PC é um dos atributos mais exigidos em *middlewares* distribuídos, mas, na prática, é raramente utilizado em grandes sistemas pois nem todos os sistemas dão suporte a 2PC. O principal problema é que todos os sistemas precisam estar funcionando e têm que fornecer os recursos até que as modificações estejam completas no último sistema. Especialmente quando há acesso concorrente ao dado, isto pode levar a atrasos e *deadlocks*.

A vantagem da compensação é que as atualizações dos sistemas não precisam ser feitas de forma síncrona (alguns sistemas podem até estar fora do ar). O ponto negativo é que precisa-se fornecer e chamar explicitamente serviços que revertam a operação realizada por um serviço anterior ou programas para manipular manualmente o erro. A linguagem BPEL suporta compensações e pode ser utilizada quando for necessário criar mecanismos para compensar operações de serviços.

4.1.2.7 Controle da lógica do processo

`Heurística: Em ambientes controlados, onde se ter um controlador central não causa problemas de gargalo, utilizar orquestração de serviços para composição de serviços. Caso o controlador central esteja impactando o desempenho da composição, utilize coreografia de serviços.`

Decisões de controle do processo podem também levar a diferentes formas de acoplamento. Ter um componente central controlando toda a lógica do processo cria um gargalo, pois cada sistema envolvido precisa se conectar com esse componente. Falhas neste componente central poderão parar todo o processo.

Por outro lado, ao se ter um controle descentralizado ou distribuído (onde cada componente realiza seu trabalho e conhece quais componentes continuarão depois) evita-se gargalos e, se algum sistema falhar, outros podem continuar trabalhando.

Normalmente, segundo Josuttis [2007], SOA é entendido como um conceito arquitetural onde os processos de negócio são projetados usando orquestração. De fato, a linguagem BPEL é uma linguagem usada puramente para orquestração de serviços e a capacidade de ser composto é quase sempre considerada como um atributo fundamental dos serviços. Todavia, a coreografia é uma questão a ser discutida. Existem padrões para serviços web que tratam de coreografia. Por outro lado, a abordagem de ter um único controlador de processo, como no caso da orquestração, tem suas limitações. A coreografia lida com uma visão mais global do sistema, aceitando que o controle possa ser distribuído entre diferentes parceiros. A utilização de coreografia faz com que o projeto dos serviços considere mensagens unidirecionais (*one-way messages*) ou eventos, o que, por sua vez, nos leva a considerar o conceito de Arquitetura Orientada a Eventos (EDA – *Event-Driven Architecture*), a qual não será discutida nesse relatório.

Em resumo, segundo Josuttis [2007], a coreografia de serviços é muito útil e promissora, mas ainda é necessário que o tempo diga quais ferramentas serão utilizadas para dar suporte à coreografia de serviços e quais terminologias deverão ser usadas. Por exemplo, não é uma opção prática.

4.1.2.8 Versionamento

`Heurística: Utilizar transformação de mensagens no barramento para realizar um versionamento implícito dos tipos de dados.`

O versionamento de serviços permite que uma nova versão de um serviço provedor não afete o funcionamento de um consumidor antigo. Contudo, quando o serviço provê algum tipo de dado que será usado por um consumidor, a atualização desse tipo de dado vai influenciar no funcionamento do consumidor e, em alguns casos, exigir que o consumidor *recompile* seu código em relação ao novo tipo de dado.

Se o provedor introduz uma nova versão de um tipo (adicionando novos atributos, por exemplo), o consumidor terá que se atualizar explicitamente. Caso contrário, o provedor terá que dar suporte às duas versões. Se, por outro lado, o provedor apenas adicionar os atributos para o tipo de dados existente, isto pode causar problemas de compatibilidade binária e o consumidor pode ter que recompilar seu código ou usar outra biblioteca.

Com uma abordagem mais fracamente acoplada de versionamento de tipos, o consumidor não terá que fazer nada enquanto as modificações mantenham a compatibilidade. Contudo, conseguir o fraco acoplamento neste quesito pode ser muito complicado. A questão aqui do versionamento de tipos de dados recai em três abordagens:

(1) Utilizar tipos diferentes para cada versão do tipo de dado: Uma nova versão de um tipo pode ser considerada como um novo tipo de dado. Regras de nomenclatura podem ser utilizadas para nomear versões de tipos, como Endereço_1 e Endereço_2. Por conseqüência, os provedores precisam usar diferentes tipos de dados para fornecer o mesmo tipo de informação e pode ser necessário que mantenham versões de um mesmo consumidor, uma nova versão do tipo e uma versão mais antiga.

(2) Utilizar o mesmo tipo de dado para diferentes versões do tipo: Neste caso o tipo precisa conter todos os atributos de todas as versões de um mesmo tipo de dado. Todos os serviços irão usar o mesmo tipo, mas eles irão usar apenas os atributos que estão especificados para eles. Isto pode se complicar bastante, pois é preciso documentar quais atributos são válidos para cada versão do serviço e nem sempre diferentes versões serão compatíveis binariamente. Ou seja, pode ser preciso ter certeza que todas as bibliotecas de um processo são compiladas com a mesma versão de um tipo. Outra questão é que se for validar os dados de entrada de acordo considerando determinado tipo, é preciso ter certeza que os atributos adicionais não vão invalidar os dados de entrada.

(3) Utilizar tipos de dados genéricos: Outra opção de lidar com diferentes versões de tipos de dados é usar apenas código genérico que seja capaz de lidar com qualquer tipo de dado. Neste caso, o formato binário da abordagem anterior não é impactante. Como conseqüência, tipos de dados não são importantes em tempo de compilação; todo o processamento acontece em tempo de execução. Isto pode ser conseguido utilizando-se códigos genéricos como `data.getValueAsString("street")`. Contudo, essa é uma abordagem muito incomum na prática e pode ser difícil dar suporte para essa abordagem em todas as plataformas de um ambiente SOA.

Segundo Josuttis [2007], uma quarta opção pode ser utilizar listas planas de parâmetros ao invés de dados não estruturados, mas essa opção não funciona quando serviços de alta granularidade enviam dados complexos.

A opção de utilizar apenas tipos de dados genéricos quase não é usada na prática e a opção de compartilhar tipos para diferentes versões é muito perigosa, uma vez que, se o processo não garante que todas as bibliotecas são consistentes (o que é difícil num sistema distribuído), pode resultar em erros e comportamento imprevisto em tempo de execução. Como consumidor de serviço, é uma boa idéia fazer o código independente do versionamento dos serviços chamados. Deveria-se usar seus próprios tipos de dados, os quais estão mapeados para o tipo de dado dos serviços quem eles forem usados. Por essa razão, os consumidores de serviço deveriam ter uma camada superior que mapeie tipos de dados externos para tipos internos (os quais podem ou não ser combinados com a camada de mapeamento entre APIs e protocolos de serviços). Contudo, segundo Josuttis [2007], também é importante manter o ambiente simples. Não vale a pena deixar o ambiente muito complexo para tentar prepará-lo para eventos que podem acontecer no futuro.

4.1.2.9 Especificação

Heurística: Separar a especificação do serviço da sua implementação, descrevendo um modelo técnico (contrato) e um modelo conceitual (requisitos, restrições etc).

Heurística: Atualizar especificação do serviço com detalhamentos da fase de projeto.

É importante ter todas as especificações dos serviços de forma independente de uma implementação particular. Hewitt [2009] aponta que WSDL é uma forma de descrever funcionalidades dos serviços independente de tecnologia. WSDL pode usualmente usar SOAP sobre http, mas isto não é um requisito; outros protocolos pode ser usados para ligação (*binding*), e o mesmo WSDL abstrato pode endereçar uma variedade de ligações, oferecendo ao mesmo tempo grande flexibilidade.

Além disso, o contrato do serviço deveria estar no nível de especificação (não no nível de implementação). Para tal, é necessário que a especificação do serviço seja criada numa ferramenta apropriada (diagramas nas ferramentas ARIS, Rational Rose, etc). Para garantir uma rastreabilidade através dos processos de negócio, é interessante relacionar a especificação do serviço com a modelagem do processo de negócio o qual o serviço apóia. Uma abordagem para descrição de serviços web utilizando a ferramenta ARIS foi proposta em [Souza *et al.*, 2009].

Heurística: Descrever formalmente a especificação do serviço.

Ao usar ferramentas, descreva formalmente a especificação do serviço tanto em um formato que seja compreensível por máquinas, para que os geradores de código possam ajudar com a implementação do serviço, como também por humanos (por exemplo, WSDL e WS-Policy). A utilização de *profiles* UML [OMG, 2009a] pode ajudar a especificar diagramas que possam ser processados por um gerador de código na linguagem utilizada pelo programador.

Dado que um serviço não é meramente um pedaço de funcionalidade, mas algo descrito por um contrato que restringe consumidores, é importante que seja definida uma interface clara dentro de um contrato. Sem uma interface clara, oportunidade de reuso de serviço é perdida. Por outro lado, é necessário pagar um preço para obter independência de plataforma, em relação a desempenho e complexidade.

4.1.3 Reuso

Heurística: Prover nomes significativos e bem descritivos do serviço e suas operações baseado no domínio do negócio, não na tecnologia. Definir padrões de nomes e descrições de acordo com as diretrizes da organização.

Heurística: Publicar descrição dos serviços em um repositório de contratos.

Heurística: Projetar serviços que possam ter uma demanda maior que a esperada.

Heurística: Serviços semelhantes devem ser consolidados em um serviço. A assinatura do serviço deve ser definida de acordo com a função mais genérica. Se as funções consideram diferentes requisitos não funcionais, então deve ser considerado o requisito funcional mais restritivo.

O reuso é o princípio mais importante no projeto de serviços. Normalmente, têm-se requisitos específicos de consumidores específicos. Contudo, quando se quer ter todas as vantagens de SOA, é necessário que os serviços que foram projetados sejam reusados por outros consumidores com requisitos ligeiramente diferentes. Ou seja, não se conhece de antemão todos os consumidores de um serviço e seus requisitos durante o projeto do serviço, o que torna a tarefa muito difícil.

4.1.4 Ausência de estado

Heurística: Evitar criação de serviços que guardam estados. Se necessário guardar estados, utilize a linguagem BPEL para tal.

Estado transacional significa que um serviço necessita ter informação sobre coisas que aconteceram como parte de uma transação de longa duração durante uma instância específica de um provedor de serviço e uma instância específica de um consumidor de serviço. Por exemplo, o resultado da invocação de uma operação de um serviço é diferente se a operação já tiver sido invocada, ou outra operação necessita ser invocada antes de outra operação. Ainda que possa ser encontrado no desenvolvimento baseado em componentes, estados transacionais devem ser evitados em SOA.

Estado da informação significa que os estados de instâncias de dados necessitam ser gerenciados. Alguns serviços normalmente necessitam gerenciar o estado dos dados. Tome por exemplo um serviço de indenização na indústria de seguros. O serviço tem que gerenciar o estado das instâncias de Indenização, especialmente porque estas instâncias são acessadas por diferentes consumidores em diferentes transações do negócio. Note que serviços que guardam estado são tipicamente atômicos (com granularidade fina). Para resolver os requisitos de estado, deve-se lidar com a tecnologia usada para implementação do serviço, como Java Plataform, Enterprise Edition (Java EE) com *stateful session beans*.

Apesar de serviços que gerenciam estados poderem ser disponibilizados, a criação dos mesmos deve ser evitada a fim de se garantir escalabilidade, independência entre instâncias do serviço, balanceamento de carga etc. Segundo [Endrei *et al.*, 2004], a linguagem BPEL deve ser utilizada para guardar estados de serviços.

4.1.5 Monitoramento de serviços

`Heurística: Definir métricas para monitoramento de serviços e utilizar ferramenta de monitoramento de serviços.`

Hewitt [2009] aponta que outra questão importante é a dificuldade em monitorar a quantidade de tráfego que um serviço está recebendo e de onde se não existe um software e métricas de monitoramento de SLA (*Service-Level Agreement*). Se o serviço é utilizado por diferentes clientes de diferentes domínios, é importante particularmente considerar que ele poderá ocorrer uma explosão de tráfego de forma repentina e inesperada devido a composição realizada por clientes fora do seu projeto. Outro problema com invocação direta é que ela adiciona complexidade na sua topografia sem dar visibilidade através de ferramentas. Existem ferramentas que demonstram a dependência através de recursos relacionados a serviços armazenados no repositório da organização. Contudo estas tecnologias podem ser caras.

4.1.6 Granularidade

`Heurística: Projetar serviços com granularidade baixa para disponibilizar serviços com alto valor ao negócio.`

Serviços são abstrações que escondem dos consumidores os detalhes de implementação. Mas um preço é pago por essa abstração, segundo Josuttis [2007]: um tempo de execução mais lento. Por exemplo, trocar chamadas de *stored procedures* para chamadas de serviços pode diminuir o tempo de acesso ao dado de um fator entre 5 e 10. Por essa razão, geralmente é melhor ter uma chamada de serviço transferindo todos os dados necessários entre um provedor e seu consumidor ao invés de ter múltiplas chamadas de serviço processando a mesma quantidade de dados.

Serviços com granularidade grossa ajudam a separar a estrutura interna de um provedor de serviço da sua interface externa. Ter um serviço para cada um acessar um atributo de uma entidade (por exemplo, um serviço para cada *setter* e *getter*) pode resultar em objetos distribuídos que aumentam as dependências entre os sistemas distribuídos.

Algumas definições de serviço, segundo Josuttis [2007], recomendam que serviços deveriam sempre ter uma granularidade grossa, mas essa recomendação tem seus problemas. Um problema é como identificar se um serviço possui a granularidade adequada. Segundo o Modelo de Referência da OASIS [OASIS, 2006], o termo "baixo acoplamento" e "granularidade grossa" não têm sido discutidos com profundidade pelo consórcio, pois "são subjetivos e não possuem métricas aplicáveis". Ainda, segundo o consórcio, "em termos de necessidades e capacidades, a granularidade geralmente está associada ao detalhamento do nível do problema que está sendo resolvido, ou seja, projetar num nível mais estratégico versus num nível algorítmico, e definir o nível ótimo não está relacionado em contar o número de interfaces ou o número de tipos de trocas de informação que estão conectadas a uma interface". Outro problema com a recomendação de serviços grossos tem a ver com considerações de desempenho. Processar uma quantidade de dados demanda tempo e, se o consumidor não precisa desses dados, o tempo empregado para o processamento será em vão.

Contudo, algumas boas práticas podem ser aplicadas referentes à granularidade dos serviços. Para o projeto orientado a serviço, deve-se pensar na granularidade no nível do provedor (quantos serviços são providos) ou no nível da especificação do serviço, por exemplo. Uma especificação de serviço é um agrupamento lógico das operações do serviço ao se considerar o negócio. Por exemplo, uma interface *Processamento de Indenizações* no domínio de seguros fornece todas as operações necessárias no cenário de uso de processamento de indenizações. Isto também faz sentido quando a implementação é levada em consideração. Por exemplo, todas as operações de um serviço seriam implementadas na mesma interação da fase de desenvolvimento, dado o agrupamento lógico das mesmas.

Quando as operações dos serviços são projetadas, considere colaborações, cenários de uso e mensagens (número e tamanho delas) que irão trafegar entre provedores e consumidores de serviços. Operações com granularidade grossa fornecem mais valor de negócio e permitem modificações mais fáceis no código. Os parâmetros desse tipo de operação (que usam mensagens para entrada, saída e erro) são menos suscetíveis a erro. Usar parâmetros com granularidade fina ao invés de mensagens, contudo, é geralmente mais descritivo. Por exemplo, se a assinatura da operação de um serviço é determineEligibility(application: ApplicationMessage), então é necessário verificar a definição de ApplicationMessage. Se a assinatura for determineEligibility(customer : Customer, product : Product, date : Date, amount : Amount), é mais descritiva. Além disso, os tipos de parâmetros Customer ou Product, por exemplo, podem ser reutilizados em outras operações, o que não é provável com ApplicationMessage.

Para a granularidade do serviço, tenha em mente que serviços podem ser compostos. Isto está relacionado com a reutilização e a habilidade de prover novas funcionalidades baseadas em funcionalidades existentes. Um conjunto de serviços em certo nível de granularidade pode ser coreografado e o resultado pode ser outro serviço com granularidade mais grossa. Pense, contudo, em ter tarefas de negócio

fornecidas pelos serviços e então ter uma sequência dessas tarefas (um processo de negócio) fornecida também como um serviço.

Por fim, tomar decisões de projeto sobre granularidade de serviço é sempre um *tradeoff* entre desempenho (por exemplo, tamanho e número de mensagens sendo trocadas e a eficiência da implementação) e potencial de reuso. Além disso, arquiteturas típicas SOA contêm tanto serviços com granularidade fina (serviços atômicos) quanto grossa (serviços compostos).

4.1.7 Resumo das heurísticas baseadas em princípios de projeto

A Tabela 12 apresenta todas as heurísticas baseadas em princípios de projeto propostas neste relatório.

Tabela 12 – Heurísticas baseadas em princípios de projeto

Princípio	Descrição
Comunicação assíncrona	Para cada serviço, priorizar implementação utilizando comunicação assíncrona. Se não houver restrição de desempenho ou a utilização do padrão assíncrono levar a uma implementação muito complexa, utilizar padrão síncrono, justificando.
Tipo de dados heterogêneos	Antes de criar um tipo de dado na implementação do serviço, verificar os tipos de dados existentes no modelo canônico, caso exista uma diretriz organizacional para sua definição.
	Não definir tipos de dados homogêneos para todos os contratos dos serviços.
Mediadores	Utilizar mediadores para invocação de serviços.
Verificação de tipos	Utilizar verificação de tipos nos mediadores quando contrato dos serviços estiver estável.
Padrão de interação	Utilizar tipos de dados comuns em diversas linguagens de programação.
Controle de transação	Definir operações de compensação para as operações do serviço.
Controle da lógica do processo	Em ambientes controlados, onde se ter um controlador central não causa problemas de gargalo, utilizar orquestração de serviços para composição de serviços. Caso o controlador central esteja impactando o desempenho da composição, utilize coreografia de serviços.
Versionamento	Utilizar transformação de mensagens no barramento para realizar um versionamento implícito dos tipos de dados.
Especificação	Separar a especificação do serviço da sua implementação, descrevendo um modelo técnico (contrato) e um modelo conceitual (requisitos, restrições, etc).
	Descrever formalmente a especificação do serviço.
Reuso	Prover nomes significativos e bem descritivos do serviço e suas operações baseado no domínio do negócio, não na tecnologia.
	Publicar descrição dos serviços em um repositório

Princípio	Descrição
	de contratos.
	Projetar serviços que possam ter uma demanda maior que a esperada.
	Serviços semelhantes devem ser consolidados em um serviço. A assinatura do serviço deve ser definida de acordo com a função mais genérica. Se as funções consideram diferentes requisitos não funcionais, então deve ser considerado o requisito funcional mais restritivo.
	Todas as funções atualizando ou modificando uma entidade (ou seja, operações de CRUD) devem ser agrupadas em um bloco de funções relacionadas à entidade.
Ausência de estado	Evite criação de serviços que guardam estados. Se necessário guardar estados, utilize a linguagem BPEL para tal.
Granularidade	Verificar se as informações processadas por um serviço estão de acordo com as informações esperadas pelos consumidores.
	Projetar serviços com granularidade baixa para disponibilizar serviços com alto valor ao negócio.

4.2 Outras boas práticas

Esta seção apresenta outras boas práticas que podem ser utilizadas em projetos de serviços para iniciativas SOA. Estas boas práticas geradas à partir da compilação de artigos avaliados no projeto [Zimmermann *et al.,* 2004a, Endrei *et al.,* 2004].

- Selecione estilos de mensageria baseado nas características de interoperabilidade e suporte ferramental: o estilo e o uso de atributos na especificação WSDL podem ser definidos como document/literal ou rpc/encoded. Historicamente, rpc/encoded possui melhor suporte ferramental e a interoperabilidade pode ser alcançada na maioria dos casos. Contudo, dado as várias ambigüidades pertencentes às regras de deserialização para rpc/encoded, o WS-I proíbe este estilo. Assim, o estilo document/literal se tornou o estilo suportado por muitas ferramentas.
- Avalie cuidadosamente quais serviços se encaixam com suas necessidades: Usar UDDI na Web é problemático não só por razões técnicas, mas também organizacionais. Questões como o modelo de negócio, qualidade dos dados e confiança necessitam ser respondidas. Por essas razões, acredita-se que UDDI é mais útil na intranet e extranet onde grupos de usuários são bem conhecidos.
- Aplique padrões pragmaticamente: Não é necessário usar sempre todos os elementos de uma tecnologia. Além disso, recomenda-se atualizar os níveis de especificação apenas se existe uma necessidade concreta para isso, e não por sua própria vontade (por exemplo, mesmo em uma segunda versão, decidir continuar com SOAP 1.1 ao invés de SOAP 1.2). Existem vários padrões de segunda geração de serviços (WS-*). Os padrões devem ser usados conforme as necessidades do projeto.

A tabela seguinte apresenta um resumo das boas práticas em projetos de serviços propostos em [Endrei *et al.,* 2004].

Tabela 13 – Boas práticas em projetos de serviços

Tópico	Explicação	Razão
Desenvolver incrementalmente	Comece com um pequeno conjunto de serviços e os construa conforme descrito nos requisitos.	Permite que todas as tecnologias sejam absorvidas, que suas próprias boas práticas possam ser implementadas e testadas.
Verificar WSDL gerado automaticamente	Use ferramentas para gerar código WSDL e verifique o WSDL gerado para garantir que ele está de acordo com os seus requisitos.	Ferramentas não podem predizer exatamente como você planeja que um serviço seja usado. Verifique o código gerado.
Estilo de ligação (*binding*)	Selecione o estilo RPC quando estiver usando um modelo RPC e a aplicação de destino não aceita XML.	Use o estilo *document* em outras situações. Use o estilo apropriado para a aplicação. O estilo RPC pode ter uma execução mais lenta que o estilo *document*.
Caching	Crie serviços de *cache* de localidades e outras informações (que não mudam com freqüência) no consumidor. O modelo de programação JAX-RPC inclui um ponto de acesso (*cached endpoint*) que poderia ser utilizado.	Minimiza a latência da aplicação quanto ao transporte de rede.
Minimizar o tamanho das mensagens XML	Evite passar dados desnecessários e considere usar tanto referências e *caching*.	Quanto maior e mais complexa a mensagem XML que é transmitida entre serviços, mais lento o serviço irá realizar usa operação.
Namepaces únicos	Utilize o nome da sua empresa ou o nome de um domínio único para evitar conflitos de *namespace*.	Garantir que os consumidores do seu serviço utilizem o esquema correto.

5 Conclusões

A especificação de serviços é o ponto principal da atividade de modelagem orientada a serviços e foca na elaboração do projeto detalhado de serviços, fluxos e componentes [Jamshidi *et al.,* 2009]. Papazoglou [2008] apresenta que a modelagem de serviços deve tratar, no mínimo, as quatro partes fundamentais de serviços: estrutura,

comportamento, política, vocabulário e melhores práticas. Portanto, a seleção de um método apropriado e comprovado para conduzir a análise e projeto de serviços tem caráter crítico para o sucesso de uma solução baseada em serviços.

Neste relatório, a fim de definirmos uma metodologia para análise e projeto de serviços, foram estudados diferentes trabalhos da literatura que abordam este tema, tais como: [Kolhborn *et al.*, 2009], [Marks e Bell, 2006], [Jones, 2006], [Saaty, 1990], [Hafezz *et al.*, 2002], [Jamshid *et al.*, 2008], [Byrne *et al.*, 2008], [Py *et al.*, 2009], [OMG, 2009a], [Heckel *et al.*, 2003], [López-Sanz *et al.*, 2007], [Johnston, 2005a], [Josuttis, 2007], [Portier, 2007], [Hewitt, 2009], [Krafzig *et al.*, 2004], [Endrei *et al.*, 2004], [Souza *et al.*, 2009], [OASIS, 2006], [Zimmermann *et al.*, 2004a] entre outros.

A metodologia proposta tem como entrada o resultado da execução do método de identificação de serviços a partir da modelagem de processos de negócio, proposto em [Azevedo *et al.*, 2009a; Azevedo *et al.*, 2009b; Azevedo *et al.*, 2008b].

A metodologia para análise de serviços candidatos envolve os seguintes passos: priorizar serviços candidatos; definir granularidade inicial de serviços candidatos; agrupar serviços candidatos; e, modelar serviços candidatos. Estes passos foram detalhados na seção 3 e inclui as heurísticas apresentadas na Tabela 14.

Tabela 14 – Heurísticas para análise de serviços candidatos

Etapa	Nome	Descrição
Priorizar serviços candidatos	Definir pesos de acordo com grau de reuso	Agrupar serviços candidatos para cada diferente grau de reuso encontrado e definir pesos para agrupamentos.
	Definir pesos para associações com sistemas	Agrupar serviços candidatos que estão associados a cada sistema que o implementa e definir pesos para agrupamentos.
	Aumentar pesos de em relação à atividades de múltiplas instâncias	Atribuir maior peso para serviços candidatos identificados a partir de atividades de múltiplas instâncias.
	Definir pesos para associações com demanda	Definir pesos para serviços candidatos associados aos requisitos da demanda. Definir maior peso para serviços candidatos associados aos requisitos de uma demanda com maior prioridade.
	Definir pesos para associações com papéis	Agrupar serviços candidatos que estão associados a cada papel e definir pesos para agrupamentos.
	Calcular priorização de serviços	Calcular priorização de serviços candidatos de acordo de acordo com o somatório dos pesos definidos por cada heurística de priorização.
	Priorizar serviços identificados a partir de fluxo	Priorizar serviços de fluxo de acordo com as informações levantadas para caracterização dos mesmos.
Definir granularidade inicial de serviços candidatos	Definir granularidade inicial de serviços candidatos	Elaborar matriz de granularidade de serviços candidatos e marcar serviços que têm maior ganho em disponibilizar como serviços físicos.

Etapa	Nome	Descrição
Agrupar serviços candidatos	Agrupar serviços de dados	Agrupar serviços de dados de acordo com operações CRUD realizadas sobre entidades do domínio.
	Definir canais de comunicação entre serviços de dados	Definir canais de comunicação entre grupos de serviços candidatos de acordo com operações *Read*.
	Reduzir custos de comunicação entre serviços	Projetar serviços buscando reduzir o custo de comunicação entre eles.
	Definir hierarquia de serviços de dados	Projetar hierarquia de serviços, incluindo serviços de acesso a dados (para as diferentes bases de dados) e serviço de integração (para integra os dados retornados pelos serviços de acesso a dados), quando for identificado reuso destes serviços em separado. Caso contrário, projetar um único serviço.
	Agrupar serviços de lógica	Agrupar serviços de lógica de acordo com padrões de uso na organização.
Modelar serviços candidatos	Modelar tipos de dados utilizados pelo serviço	Modelar os tipos de dados utilizados pelo serviço em um diagrama UML, associando os tipos de dados com o modelo canônico, quando este existir.
	Especificar serviços utilizando profile	Documentar a especificação dos serviços utilizando um profile UML para SOA.

No caso da proposta para projeto de serviços candidatos, foi elaborado um conjunto de heurísticas de projeto, apresentadas na Tabela 15.

Tabela 15 - Heurísticas para projeto de serviços

Heurística	Descrição
Priorizar serviços auto-contidos	Priorizar implementação de serviços auto-contidos. Caso seja necessário que um serviço invoque outro serviço, considerar o uso de uma tecnologia de orquestração de serviços.
Priorizar implementação assíncrona de serviços	Para cada serviço, priorizar implementação utilizando comunicação assíncrona. Se não houver restrição de desempenho ou a utilização do padrão assíncrono levar a uma implementação muito complexa, então utilizar padrão síncrono, mas justificando.
Definir tipos de dados do serviço baseado no modelo canônico	Antes de criar um tipo de dado na implementação do serviço, verificar os tipos de dados existentes no modelo canônico, caso exista uma diretriz organizacional para sua definição.
Não homegeinizar tipos de dados de muitos serviços	Não definir tipos de dados homogêneos para todos os contratos dos serviços.
Utilizar mediadores	Utilizar mediadores para invocação de serviços.

Heurística	Descrição
Utilizar verificação fraca de tipos	Utilizar verificação de tipos nos mediadores quando contrato dos serviços estiver estável.
Evitar tipos complexos de dados	Utilizar tipos de dados comuns em diversas linguagens de programação.
Tratar transações com compensação	Definir operações de compensação para as operações dos serviços.
Avaliar uso de orquestração e coreografia	Em ambientes controlados, onde se ter um controlador central não causa problemas de gargalo, utilizar orquestração de serviços para composição de serviços. Caso o controlador central esteja impactando o desempenho da composição, utilize coreografia de serviços.
Versionar seviços utilizando transformação de mensagens	Utilizar transformação de mensagens no barramento para realizar um versionamento implícito dos tipos de dados.
Especificar serviços em diferentes níveis de abstração	Heurística: Separar a especificação do serviço da sua implementação, descrevendo um modelo técnico (contrato) e um modelo conceitual (requisitos, restrições etc).
Nomear serviços de forma significativa e segundo padrão organizacional	Prover nomes significativos e bem descritivos do serviço e suas operações baseado no domínio do negócio, não na tecnologia. Definir padrões de nomes e descrições de acordo com as diretrizes da organização.
Utilizar repositório de serviços	Publicar descrição dos serviços em um repositório de contratos.
Generarlizar serviços para aumentar reuso	Projetar serviços que possam ter uma demanda maior que a esperada.
Agrupar serviços	Serviços semelhantes devem ser consolidados em um serviço. A assinatura do serviço deve ser definida de acordo com a função mais genérica. Se as funções consideram diferentes requisitos não funcionais, então deve ser considerado o requisito funcional mais restritivo.
Priorizar implementação de serviços sem estado	Evitar criação de serviços que guardam estados. Se necessário guardar estados, utilize a linguagem BPEL para tal.
Monitorar serviços em execução	Definir métricas para monitoramento de serviços.
Tratar granularidade do serviço	Projetar serviços com granularidade que leve o serviço a ter alto valor ao negócio.

Como trabalhos futuros, propomos uma análise das heurísticas de projeto a fim de definir uma metodologia composta de passos detalhados para projeto de serviços. Além disso, pretendemos avaliar na prática as propostas elaboradas.

Agradecimentos

Este trabalho não seria possível sem a contribuição de pesquisadores em Sistemas de Informação e da parceria com a Petrobras, principalmente a área TIC/TIC-E&P/GDIEP. Em especial, agradecemos aos professores e alunos que colaboraram nas discussões e desenvolvimento de pesquisas, testes e desenvolvimentos necessários ao projeto. Dentre os agradecimentos à academia, se destaca o papel dos profissionais do NP2Tec[2] que contribuíram, técnica ou administrativamente, para o sucesso de nossas atividades.

A condução e os resultados deste trabalho são uma exemplar evidência de como a relação entre as universidades e as empresas pode contribuir para a geração de conhecimento útil e, desta forma, contribuir para nossa sociedade.

Referências Bibliográficas

ADAM, S.; RIEGEL, N.; DOERR, J. **Deriving Software Services from Business Processes of Representative Customer Organizations**. In Service-Oriented Computing: Consequences for Engineering Requirements, SOCCER '08. International Workshop on. pp. 38-45, 2008.

ALONSO, G.; **Web Services: Concepts, Architectures, and Applications**. Springer, 2004.

AMSDEN, J. **Modeling SOA: Part 1. Service specification**. IBM developerWorks, 2007.

ARSANJANI, A. et al.. **SOMA: A method for developing service-oriented solutions**. IBM Systems Journal, 47(3), 2008.

ARSANJANI, A.. **Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA**, 2004. Disponível em <http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/>. Acesso em: 01 mar. 2009.

AZEVEDO, L. G.; SANTORO, F. M.; BAIAO, F.; SOUZA, J. F.; REVOREDO, K.; PEREIRA, V. B.; HERLAIN, I.. **A Method for Service Identification from Business Process Models in a SOA Approach**. In: 10th International Workshop on Business Process Modeling, Development, and Support (BPMDS), Amsterdam, 2009a.

AZEVEDO, L. G.; BAIAO, F.; SANTORO, F. M.; SOUZA, J. F.; REVOREDO, K.; PEREIRA, V. B.; HERLAIN, I. **Identificação de serviços a partir da modelagem de processos de negócio**. In: Simpósio Brasileiro de Sistemas de Informação (SBSI), Brasília, 2009b.

[2] Site do NP2Tec: http://www.uniriotec.br/~np2tec

AZEVEDO, L.; PEREIRA, V.; BAIAO, F.; SANTORO, F.; SOUZA, J.; REVOREDO, K. **Relatório de Conceituação em SOA,** Pesquisa em Arquitetura Orientada a Serviços, E&P/GDIEP, Petrobras, 2008a.

AZEVEDO, L.; PEREIRA, V.; BAIAO, F.; SANTORO, F.; SOUZA, J.; REVOREDO, K. **Relatório de Metodologias de Identificação de Serviços**, Pesquisa em Arquitetura Orientada a Serviços, E&P/GDIEP, Petrobras, 2008b.

BROWN, A.W. et al. **Realizing service-oriented solutions with the IBM rational software development platform**. IBM Systems Journal, 44(4), 727-752, 2005.

BYRNE, B.; MCCARTY, D.; SAUTER, G.; WORCESTER, P.; KLING, J. **Introduction to the information perspective of a Service Oriented Architecture,** 2008c. Disponível em <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0801sauter/>. Acesso em: 10 jun. 2008.

COCKBURN, P. **Writing effective use cases**, Addison Wesley, 2001.

ENDREI, M. et al. **Patterns: Service-Oriented Architecture and Web Services**, 2004. Disponível em <www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf>.

FAREGHZADEH, N. **Service Identification Approach to SOA Development**. World Academy of Science, Engineering and Technology, 35, 2008.

HAFEEZ, K.; ZHANG, Y.B.; MALAK, N. **Determining Key Capabilities of a Firm Using Analytic Hierarchy Process**, Int'l J. Production Economics, vol. 76, no. 1, pp. 39-51, 2002.

HECKEL, R.; LOHMANN, M.; THÖNE, S. **Towards a UML Profile for Service-Oriented Architectures**, Workshop on Model Driven Architecture: Foundations and Applications, 2003.

HEWITT, E. **Java SOA Cookbook**, O'Reilly, 2009.

INAGANTI, S., BEHARA. G. K. **Service Identification: BPM and SOA Handshake**, Technical Report, Business Process Trends, 2007.

JAMSHIDI, P. et al. **An automated method for service specification**. In Proceedings of the Warm Up Workshop for ACM/IEEE ICSE 2010. Cape Town, South Africa: ACM, pp. 25-28, 2009. Disponível em <http://portal.acm.org/citation.cfm?id=1527042&dl=ACM>. Acesso em: 01 mai. 2009.

JAMSHIDI, P.; SHARIFI, M.; MANSOUR, S. **To Establish Enterprise Service Model from Enterprise Business Model**. IEEE International Conference on Services Computing, 1, 93-100, 2008.

JOHNSTON, S. **UML 2.0 Profile for Software Services,** 2005a. Disponível em <http://www.ibm.com/developerworks/rational/library/05/419_soa>.

JOHNSON, S. **Modeling service-oriented solutions**, 2005b. Disponível em <http://www.ibm.com/developerworks/rational/library/jul05/johnston>.

JOHNSTON, S.; SMITH, J. **RUP Plug-In for SOA V1.0**. IBM developerWorks, 2005.

JONES, S. **Enterprise SOA Adoption Strategies**. Using SOA to Deliver IT to the Business. C4Media, 2006.

JOSUTTIS, N. M. **SOA in practice: The Art of Distributed System Design**, O'Reilly, 2007.

KLOSE, K; KNACKSTEDT, R.; BEVERUNGEN, D. **Identification of Services – A Stakholder-Bases Approach to SOA Development and its Application in the Area of Production Planning**, Proc. 15th European Conf. Information Systems (ECIS'07), pp. 1802-1814, 2007.

KOHLBORN, T.; KORTHAUS, A.; CHAN, T.; ROSEMMAN, M. **Identification and Analysis of Business and Software Services – A Consolidated Approach**, IEEE Transactions on Service Computing, Vol 2, No 1, 2009.

KOHLMANN, F.; ALT, R. **Business-Driven Service Modelling - a Methodological Approach from the Finance Industry**. Proc. First Int.'l Working Conference Business Process and Services Computing (BPSC'07), pp. 180-193, 2007.

KRAFZIG, D.; BANKE, K.; SLAMA, D. **Enterprise SOA: Service-Oriented Architecture Best Practices**. Upper Saddle River, NJ: Prentice Hall, 2004.

LÓPEZ-SANZ, M.; ACUÑA, C.J.; CUESTA, C.E.; MARCOS, E. **UML Profile for the Platform Independent Modelling of Service-Oriented Architectures**, First European Conference, ECSA 2007 Aranjuez, Spain, September 24-26, 2007.

MARKS, E. A.; BELL, M. **Service-Oriented Architecture: a planning and implementation guide for business and techonology**, John Willey & Sons Inc, 2006.

MCSHEFFREY, S.D. **Integrating Business Process Models with UML System Models**, Technical Report, Popkin Software, 2001.

MODELDRIVEN. **Ferramenta ModelPro**, 2009. Disponível em <http://portal.modeldriven.org/project/modelpro>.

OASIS. **Reference Model for Service Oriented Architecture**, Committee Draft 1.0, 2006. Disponível em <http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf>.

OMG. **Catalog of UML Profile Specifications**, 2009a. Disponível em <http://www.omg.org/technology/documents/profile_catalog.htm>.

OMG. **Service-oriented Architecture Modeling Language (SoaML),** 2009b. Disponível em <http://www.omg.org/docs/ad/08-08-04.pdf>.

PAPAZOGLOU, M. P. **Web Services: Principles and Technology**, Prentice Hall, 782 pp, 2007.

PAPAZOGLOU, M. P.; TRAVERSO, P.; DUSTDAR S.; LEYMANN F. **Service-Oriented Computing: a Research Roadmap**. Proceeding of Int. J. Cooperative Inf. Syst. 17(2): 223-255, 2008.

PORTIER, B. **SOA terminology overview, Part 3: Analysis and design**, 2007. Disponível em <http://www.ibm.com/developerworks/webservices/library/ws-soa-term3/>.

PRESSMAN, R. **Software Engineering: A Practitioner's Approach**, 6 ed., McGraw-Hill, 2004.

PY, H.; CASTRO, L.; BAIÃO, F.; TANAKA, A. **A Service-Based Approach for Data Integration Based on Business Process Models**. In: International Conference on Enterprise Information Systems, p. 222-227, Milan, 2009.

SAATY, T., **Decision Making for Leaders**. The Analytical Hierarchy Process for Decisions in a Complex World. RWS Publications, 1990.

SANZ, J.; NAYAK, N.; BECKER, V. **Business Services as a Modeling Approach for Smart Business Networks**, Technical Report RJ10381 (A0606-001), IBM Research Division, Almaden Research Center, 2006.

SOL, H.G. **Information Systems Development: A Problem Solving Approach**, Proceedings of the International Symposium on System Development Methodologies, 1990.

SOUZA, J. F.; AZEVEDO, L. G.; SANTORO, F.; BAIÃO, F. **Relatório de modelagem de serviços**. Pesquisa em Arquitetura Orientada a Serviços, E&P/GDIEP, Petrobras, 2009.

STOJANOVIĆ, Z. **A Method for Component-Based and Service-Oriented Software Systems Engineering**, Doctoral Thesis, Delft University of Technology, Netherlands, ISBN: 90-9019100-3, 2005.

TEALE, P.H.; JARVIS, R. **Business Patterns for Software Engineering Use**, The Architecture Journal, 2004. Disponível em <http://msdn2.microsoft.com/enus/arcjournal/aa480036.aspx>.

ZHANG, Z.H.; LIU, R.; YANG H. **Service Identification and packaging in Service-oriented Reengineering**, 2005. Disponível em <http://www.tech.dmu.ac.uk/STRL/research/publications/pubs/2005/2005-8.pdf>.

ZIMMERMANN, O.; KOEHLER, J.; LEYMANN, F. **Architectural decision models as micro-methodology for Service-Oriented Analysis and Design**, Proceedings of the SEMSOA Workshop on Software Engineering Methods for Service-Oriented Architecture, 46-60, 2007.

ZIMMERMANN, O.; KROGDAHL, P.; GEE, C. **Elements of Service-Oriented Analysis and Design: An interdisciplinary modeling approach for SOA projects**, 2004a. Disponível em <http://www.ibm.com/developerworks/webservices/library/ws-soad1/>.

ZIMMERMANN, O. et al. **Second generation web services-oriented architecture in production in the finance industry**. In Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications. Vancouver, BC, CANADA: ACM, pp. 283-289, 2004b. Disponível em <http://portal.acm.org/citation.cfm?id=1028772&coll=GUIDE&dl=GUIDE&CFID=1860570&CFTOKEN=81006190&ret=1>. Acesso em: 5 abr. 2009.

Anexo A: Propostas existentes para análise e projeto de serviços

Neste capítulo são apresentadas as metodologias propostas na literatura que tratam da análise e projeto de serviços.

A.1 SOMA (Service-Oriented Modeling and Architecture)

A.1.1 Descrição da proposta

Arsanjani [2004] propõe SOMA (*Service-Oriented Modeling and Architecture*) que corresponde a uma metodologia para implantação de abordagem SOA em uma organização. Arsanjani [2004] ressalta que a estratégia para projeto de serviços não inicia debaixo para cima ("bottom-up") como frequentemente acontece no caso das abordagens baseadas em *web services*. SOA é mais estratégico e alinhado ao negócio. *Web services* são a implementação tática de uma SOA.

Arsanjani [2004] propõe um conjunto de camadas para SOA, como apresentado na Figura 12. É preciso projetar e tomar decisões arquiteturais para cada uma das camadas e documentar a iniciativa como, por exemplo, utilizando um documento como o apresentado na Figura 12, documentando em seções cada uma das 7 camadas propostas.

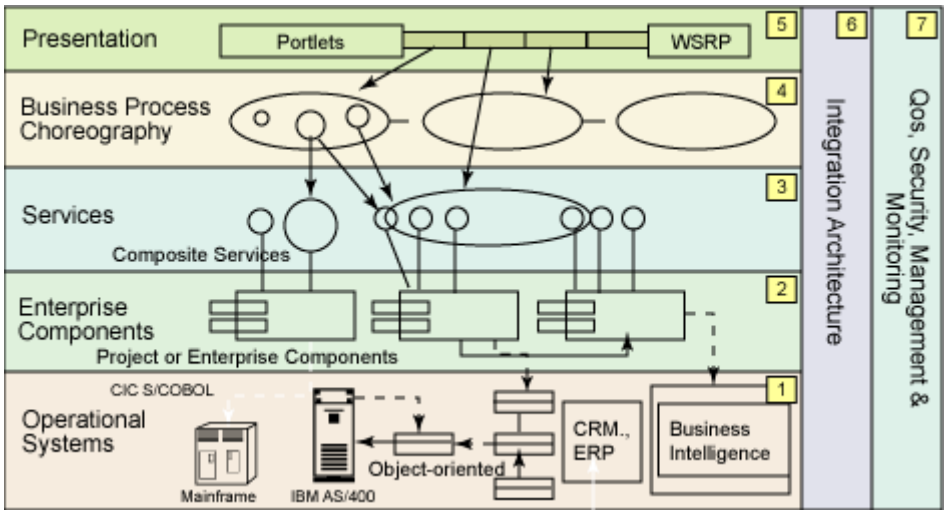


Figura 12 – Camadas para uma SOA

A camada de sistemas operacionais consiste de sistemas legados. A camada de componentes empresariais é a responsável por prover a funcionalidade e manter a qualidade de serviço (QoS) dos serviços disponíveis. Esta camada normalmente usa tecnologias baseadas em containers como servidores de aplicação para implementar componentes, gerenciamento de carga, alta disponibilidade. A camada de serviços externaliza as interfaces dos ativos da empresa no formato de descrições de serviços. A camada de coreografia ou de composição de processos é onde os serviços compostos

estão dispostos. Nesta camada, softwares para modelagem de fluxos podem ser usados para compor os serviços. A camada de acesso ou apresentação, por sua vez, está ganhando relevância, pois é onde a convergência de padrões como WSDL, Portlets e WSRP permitem novos canais de acesso para os serviços existentes. A camada de integração é normalmente descrita como um ESB que provê funcionalidades como roteamento inteligente, mecanismos de transformação, protocolos de mediação, etc. Por fim, a camada de QoS provê as capacidades necessárias para monitorar, gerenciar e manter QoS como segurança, desempenho, disponibilidade, etc.

1. Escopo ("Para quais áreas da empresa é esta arquitetura?")
2. Camada de sistemas operacionais
   - 2.1. Aplicações empacotadas
   - 2.2. Aplicações customizadas
   - 2.3. Decisões arquiteturais
3. Camada de componentes da empresa
   - 3.1. Áreas funcionais apoiadas pelos componentes da empresa
   - 3.2. ("Quais domínios de negócio, objetivos e processos são apoiados pelos componentes da empresa?)
   - 3.3. Decisões de governança
     - 3.3.1. ("Critério pelo qual um componente é eleito como um componente da empresa dentro de uma organização cliente")
   - 3.4. Decisões arquiteturais
4. Camada de serviços
   - 4.1. Portfólio categorizado de serviços
   - 4.2. Decisões arquiteturais
5. Camada de processos de negócio e de composição
   - 5.1. Processos de negócio a serem coreografados
   - 5.2. Decisões arquiteturais
     - 5.2.1. "Quais processos de negócio serão coreografados em serviços e quais processos de negócio serão implementados dentro de aplicações?"
6. Camada de acesso ou representação
   - 6.1. ("Implicações de documento de web services e SOA nesta camada, se existir. Por exemplo, uso de portlets que invocam web services na interface do nível do usuário e as implicações no funcionamento da camada.")
7. Camada de integração
   - 7.1. <"Incluir considerações de um ESB")
   - 7.2. <"Como estamos indo para garatir SLA (Service Level Agreements) e QoS (Quality of Services) requeridos pelos clientes dos serviços providos?
   - 7.3. Decisões e questões de segurança
   - 7.4. Decisões e questões de desempenho
   - 7.5. Decisões e limitações de tecnologia e padrões
   - 7.6. Monitoramento e gestão de serviços
     - 7.6.1. Descrições e decisões.

Figura 13 - *Template* de documento para descrever a arquitetura

Um número de importantes atividades e decisões existentes influenciam não apenas a arquitetura de integração, mas também as arquiteturas da empresa e de aplicações. Elas incluem atividades desde as duas visões de consumidor e provedor, descritas na Figura 14. Observe que as atividades do provedor de serviços

correspondem a um superconjunto das atividades do consumidor de serviços. Ou seja, o provedor também tem que tratar a identificação, categorização, exposição, coreografia e qualidade de serviços. Em muitos casos, a diferenciação entre os papéis vem do fato de que o consumidor especifica os serviços que deseja, frequentemente procuram por eles, e uma vez que estão convencidos de que a especificação do serviço que estão procurando está de acordo com o serviço desejado, e que é provido por um provedor de serviço, eles se conectam e invocam o serviço. O provedor, por sua vez, necessita publicar os serviços que eles desejam disponibilizar; tanto em termos de funcionalidades como em termos de QoS (*Quality of Service* – Qualidade do Serviço) que os consumidores requerem. Este contrato implícito entre consumidor e provedor pode amadurecer em um contrato em termos de SLAs (*Service Level Agreements*); negociado tanto eletronicamente ou através de locais de negócio ou legalizados.

Papel	**Atividades neste papel**				
Visão do consumidor	Identificação do serviço	Categorização do serviço	Decisões de exposição do serviço	Coreografia ou composição	Qualidade de serviço
Visão do provedor	Identificação do componente	Especificação do componente	Realização do serviço	Gerência do serviço	Implementação de padrões
	Alocação de serviço para componentes	Definição das camadas SOA	Prototificação técnica	Seleção do produto	Decisões arquiteturais (estado, fluxo, dependências)

Figura 14 – Atividades da modelagem orientada a serviços nas visões do consumidor e provedor

As atividades descritas na Figura 14 podem ser descritas como apresentado na **Figura 15** como a metodologia SOMA, a qual foi composta de 3 fases principais: identificação, especificação e realização.

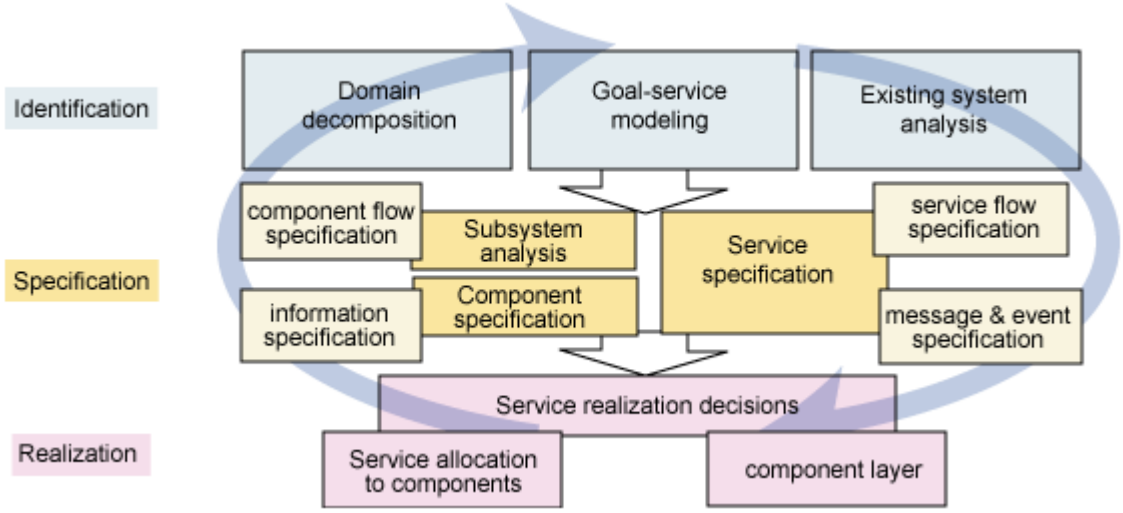


Figura 15 – Método Modelagem e Arquitetura Orientada a Serviços (SOMA)

A fase de identificação de serviços consiste uma combinação de técnicas *top-down* (decomposição do domínio em áreas funcionais e subsistemas, incluindo fluxos de

negócio), *bottom-up* (análise de ativos existentes que permite encontrar soluções de baixo custo de implementação) e *middle-out* (modelagem de objetivos, que permite ligar os serviços a objetivos, KPIs e métricas).

A atividade de classificação e categorização é importante para refletir a natureza de composição ou fractal dos serviços através de uma classificação em hierarquia. A classificação ajuda a determinar a composição. Além disso, evita a proliferação de serviços muito granulares.

A atividade de análise de subsistemas pega os subsistemas encontrados durante a decomposição do domínio e especifica as interdependências e o fluxo entre os subsistemas.

A atividade de especificação de componentes é responsável por especificar os detalhes dos componentes como dados, regras, serviços, perfis configuráveis, variações. A especificação de eventos e mensageria e definição de gerenciamento ocorrem nesse passo.

A atividade de alocação de serviços consiste em ligar serviços a subsistemas que foram identificados. Essa atividade também consiste em ligar os serviços e componentes a cada camada SOA.

Por último, a fase de realização é onde o software que provê um dado serviço será selecionado ou construído. Neste passo, é preciso decidir quais módulos dos sistemas legados serão utilizados para realizar um dado serviço e quais serviços serão construídos. Outras decisões incluem segurança, monitoramento e gerenciamento dos serviços.

A.1.2 Análise da proposta

Apesar de ser apresentado um conjunto de passos para modelagem e arquitetura de serviço, as atividades não são apresentadas em detalhes, sendo necessário um maior aprofundamento das mesmas. Além disso, a metodologia deve ser instanciada para cada projeto (onde decisões específicas de projeto devem ser tomadas). Porém, são poucos os indicativos que o autor fornece para basear decisões de projeto.

A.2 Service Identification Approach to SOA Development

A.2.1 Descrição da proposta

Neste trabalho, Fareghzadeh [2008] apresenta que apesar do conceito de SOA ser muito debatido nos últimos anos, ainda não surgiu uma metodologia unificada para identificar serviços. Ao invés disso, várias abordagens heterogêneas têm sido propostas, especialmente abordagens utilizando sistemas de informação (*bottom-up*), modelos de processos de negócio (*top-down*) ou abordagens híbridas. O objetivo dessa proposta, então, é propor uma abordagem que combina diferentes abordagens, evitando as desvantagens de algumas dessas abordagens e facilitando o passo-a-passo de identificação de serviços, sendo um guia para implementação dos serviços.

A proposta de identificação de serviços do artigo foi dividida em 3 fases: Análise Inicial, Análise em Profundidade e Criação de uma taxonomia de serviço.

A fase de Análise Inicial é responsável pela tomada de decisões básicas como o escopo do projeto, requisitos de negócio, visões e objetivos de negócio. A análise dos

serviços deve ser documentada, identificando as funções mais importantes do domínio e mostrando suas interrelações. As principais saídas dessa fase são os requisitos, objetivos e visões do negócio; modelos de processo de negócio; componentes e serviços reusáveis; glossário de termos para facilitar o vocabulário comum entre os membros da equipe.

Na fase de análise em profundidade, 2 passos são realizados. Primeiramente, deve-se pegar os processos documentados e quebrá-los em series de passos de processos granulares e independentes. Depois, deve-se associar esses passos aos papéis que irão executá-los. Esses papéis são entidades e atores do negócio. Para facilitar a identificação das funções que poderão virar serviços, o artigo relaciona algumas perguntas que devem ser feitas para identificar os serviços como, por exemplo, "A função pode ser governada?", "Quais aspectos da função estão visíveis para os *stakeholders*?", "Quais canais (internet, PDAs, etc) poderiam ser usados para enviar a informação requerida para o cliente?", etc. Uma vez feito isso, os autores propõem interrelacionar as dependências entre as funções para criar casos de uso e, assim, extrair a operações dos serviços candidatos e as regras de negócio.

No segundo passo da análise em profundidade, deve-se refinar a definição do "sistema SOA" e a análise das áreas críticas.

A última etapa consiste na criação da taxonomia dos serviços, onde autores propõem identificar os tipos de serviços existentes e agrupá-los por funcionalidade.

A.2.2 Análise da proposta

A proposta de Fareghzadeh [2008] contempla a identificação de serviços e a análise de serviços. É tratada a utilização de vários modelos como insumo para a análise de serviços candidatos e também propõe-se em lidar com dois mundos comumente distintos: as demandas da TI e as demandas do negócio. Contudo, o método para identificação de serviços candidatos apresentado é fortemente dependente da avaliação de um analista, o que apresenta grandes dificuldades para automatizar o processo.

O artigo trata mais de identificação de serviços, falando muito pouco de análise e projeto de serviços. Apresenta uma classificação diferente para serviços e um passo-a-passo para identificá-los, o qual ainda está em alto nível. Apesar do artigo falar que apresenta atividades de "Análise de serviços", estas são mais para identificar e classificar serviços. Infelizmente, o artigo não apresenta princípios que poderiam na análise do serviço candidato identificado a fim de ajudar a decidir se este deve ser implementado.

Além disso, a proposta apresenta uma lista de perguntas que poderiam ser utilizadas para identificar serviços, que podem ser analisadas para ajudar na construção de uma lista de perguntas de análise de serviços, por exemplo: quantos sistemas estão envolvidos no serviço candidato, quantas entidades estão envolvidas etc. Estas perguntas poderiam complementar a fase de consolidação de serviços de [Azevedo *et al.*, 2008].

A.3 Elements of Service-Oriented Analysis and Design: An interdisciplinary modeling approach for SOA projects

A.3.1 Descrição da proposta

Neste trabalho, Zimmermann *et al.* [2004a] apresentam uma proposta para combinação de elementos de práticas bem estabelecidas, tais como OOAD (*Object Oriented Analysis and Design*), EA (*Enterprise Architecture*), e BPM (*Business Process Modeling*), complementando-os com elementos específicos da nova da demanda.

A partir de elementos adequados de OOAD, EA, e BPM, é definida uma abordagem híbrida *Service-Oriented Analysis and Design* (SOAD), combinando elementos destas disciplinas. A abordagem incorpora novos elementos, tais como conceituação de serviço (identificação), agregação e categorização de serviços, políticas, processo *meet-in-the-middle*, semântica, e discriminação de serviços (para reutilização).

Uma tendência recente é definir formas padrão de representação executável de modelos de fluxo (por exemplo, BPEL). BPEL estende o alcance dos modelos de processo da análise para implementação. Tais modelos executáveis levam a um conjunto de novas questões:

- Quais aspectos devem ser descritos em BPEL, e quais, em WSDL? Onde está o limite entre modelo de processo e modelo de programação?
- Como aspectos não-funcionais e requisitos de qualidade podem ser incorporados nos modelos?
- Quanto de lógica de programação é executado por motores de extensões da linguagem BPEL, em oposição a tradicional codificação, por exemplo, em J2EE?
- Como pode ser avaliada a qualidade do processo executável, e quais são as melhores práticas que se aplicam?
- Que papel na empresa deve ser responsável por elaborar o fluxo BPEL; um analista de negócio especializado ou um arquiteto de software?

Os modelos de processo de negócio existentes podem ser aproveitados como ponto de partida para SOAD, contudo, têm de ser alterados, com técnicas adicionais para identificação de serviços candidatos e as suas operações. Além disso, o modelo de processo em SOAD deve ser sincronizado com design de modelagem de casos de uso e utilizar as respostas às perguntas relacionadas à BPEL.

Os seguintes requisitos foram identificados para SOAD:

- processo e notação tem de ser formalmente definidos;
- deve haver uma forma estruturada para conceituar serviços: OOAD nos dá classes e objetos no nível da aplicação, enquanto BPM tem modelos orientados eventos, SOAD deve juntar os dois; o método não é mais orientado para casos de uso, é orientado a eventos e processos, mas casos de uso aparecem na etapa seguinte; o método deve incluir sintaxe, semântica, e políticas;
- SOAD deve prover fatores de qualidade bem-definidos, e melhores práticas. Por exemplo, a questão dos papéis levantada pelo BPEL deve ser respondida;

- SOAD também terá que responder à pergunta: O que não é um bom serviço? Por exemplo: tudo o que não é, ou nunca será reutilizável, provavelmente não será um bom serviço;
- SOAD deve facilitar a modelagem *end-to-end* e ter suporte abrangente através de ferramentas de apoio.

Alguns princípios gerais ou fatores de qualidade já foram identificados como uma concepção inicial dentro de SOAD:

- Serviços bem delineados trazem flexibilidade e agilidade ao negócio, facilitam a facilidade de reconfiguração e reutilização através de acoplamento baixo e encapsulamento.
- Serviços bem concebidos são significativos e aplicáveis a mais aplicações; dependências entre serviços são minimizadas e quando existem, são explicitamente indicadas.
- Abstrações de serviços são coesas, completas e coerentes, por exemplo, deve-se pensar na metáfora CRUDS na concepção de serviços e assinaturas de suas operações.
- Serviços são desprovidos de estado, na medida do possível no domínio do problema e contexto.
- O nome do serviço é compreensível por especialistas do domínio sem profundos conhecimentos técnicos.
- Todos os serviços seguem a mesma filosofia de design (que se articula através de padrões e modelos) e padrões de interação, o estilo arquitetônico subjacente pode ser facilmente identificado.
- O desenvolvimento dos serviços e consumidores requer apenas competências básicas na linguagem de programação, além domínio conhecimento; conhecimento sobre o *middleware* só é exigido para alguns especialistas, que em um mundo ideal, trabalham para os vendedores de ferramentas, e não para as empresas que criam aplicações empresariais com SOA.

Os autores descrevem como próximos passos para elaboração da SOAD, a necessidade de definição de uma notação fim-a-fim para processo, revisão dos papéis necessários para elaboração destes modelos, e validação da proposta em projetos.

A.3.2 Análise da proposta

Os autores apresentam uma proposta de metodologia para análise e projeto de serviços em SOA baseada em um conjunto de critérios de qualidade e requisitos desejados. Os requisitos apresentados para a metodologia estão em conformidade com a nossa proposta. No entanto, não descrevem as soluções para solucionar as questões levantadas (que são bastante semelhantes àquelas levantadas no nosso projeto).

A.4 UML 2.0 Profile for Software Services

A.4.1 Descrição da proposta

Simon Johnston [2005a] apresenta um *profile* UML criado pela IBM para descrição de serviços nas ferramentas Rational. O *profile* UML, para descrição de serviços, criado pela IBM Rational é utilizado para descrever atividades de desenvolvimento do ciclo de vida do serviço e visões para diferentes *stakeholders*.

A Figura 16 apresenta o *profile* criado pela IBM utilizando os estereótipos definidos para representar elementos de descrição de serviços. A figura demonstra os detalhes do profile, apresentando cada estereótipo com sua metaclasse usando a notação de extensão (setas com ponta preenchida). As restrições estão descritas nesse modelo como notas.

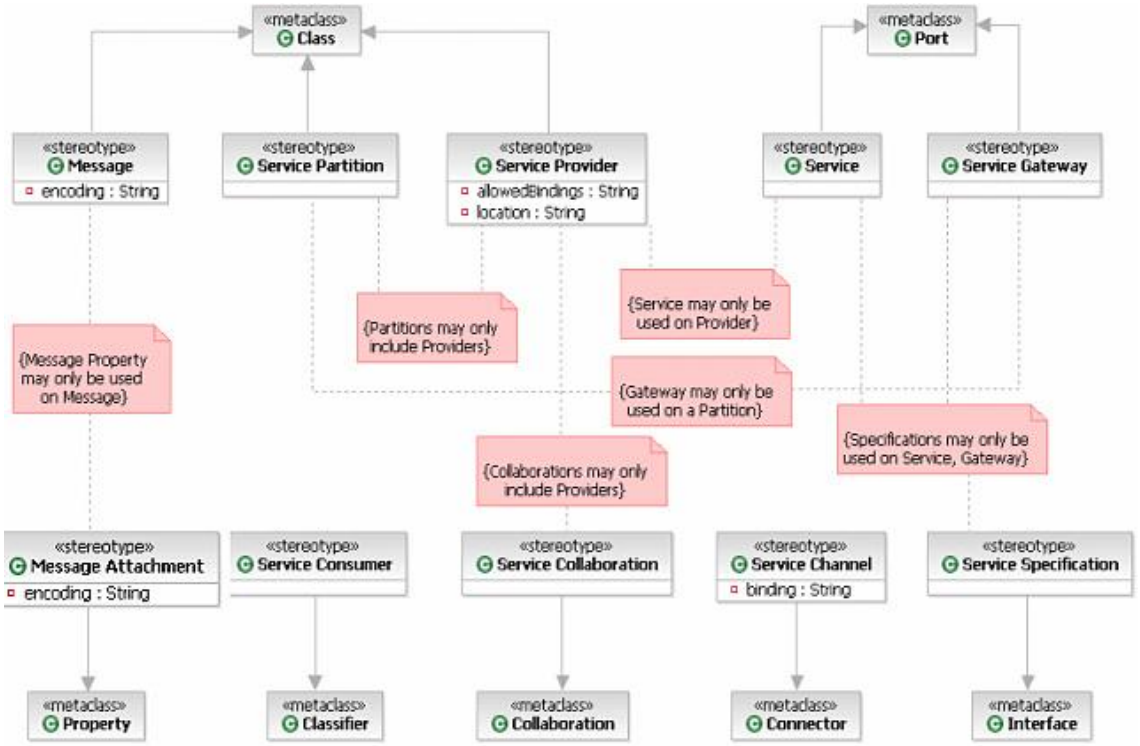


Figura 16 – *Profile* para descrição de serviços

A.4.2 Análise da proposta

É interessante a extensão da UML e é importante notar que, segundo o artigo, essa extensão está incorporada no Rational Software Architecture através de um *plug-in* e foi utilizado em vários projetos. O *plug-in* somente funciona para o RSA. Contudo, algumas outras ferramentas case permitem estender a UML. É necessário verificar se o ARIS permite fazer essa extensão caso formos modelar serviços em UML. Por outro lado, ferramentas como o ARIS apresentam um modelo UML para descrever WSDL, por exemplo. Ainda assim, o modelo da IBM é interessante, pois, no mesmo UML, permite modelar consumidores e provedores, algo que o ARIS não inclui no UML. Não obstante essa falta do ARIS, já foi feito um estudo no projeto de como apresentar essas

informações utilizando os elementos para modelagem de processos do ARIS. Então, é uma decisão mais de ferramenta do que de forma.

A.5 To Establish Enterprise Service Model from Enterprise Business Model

A.5.1 Descrição da proposta

O objetivo do artigo [Jamshidi *et al.*, 2008] é tratar aspectos dos passos iniciais para construção de soluções baseadas em serviço – modelagem de serviços, considerando modelos de negócio.

O artigo caracteriza os seguintes trabalhos relacionados:

- [Inaganti & Behara, 2007] apresenta uma proposta limitada à identificação de serviços que é clara na identificação de serviços no nível da empresa e não está limitada em apenas listar os serviços identificados. Entretanto, as medidas para identificação das atividades de negócio não é apresentada; a proposta não é validada na prática, e não é descrito nenhum notação de modelagem padrão tais como BPMN para modelagem de processos e *profiles* UML para arquitetura orientada a serviços.
- [Zhang *et al.*, 2005] propôs um processo de reengenharia orientada a serviços para reengenharia de arquitetura de software. Sua técnica adota componentes legados e os empacota em serviços identificados a partir do modelo do domínio.
- [Zimmerman *et al.*, 2004a] propõem os elementos chaves e uma nova disciplina SOAD (*Service-Oriented Architecture Analysis and Design*) para definir elementos de um modelo de serviços, mas não apresentam nenhum procedimento para construção dos modelos de serviços propostos. Eles também apresentaram um modelo em [Zimmerman *et al.*, 2007] para realização de serviços, com o principal objetivo de propor uma metodologia prescritiva para realização de serviços a fim de simplificar decisões arquiteturais.
- [Arsanji, 2004] também apresentam as atividades chaves de SOAD. Um *template* para este tipo de arquitetura é apresentado e a importância de endereçar estas técnicas é discutida. As atividades de identificação e especificação de serviços são apresentadas, mas não é dito nada a respeito de como executá-las.
- [Portier, 2007] apresentam uma taxonomia para SOAD, mas também não detalha as atividades e técnicas.
- [Johnson, 2005a; Johnson 2005b] introduziu alguns temas chaves para adaptar RUP para modelagem de serviços. Um *profile* UML para suportar a notação da modelagem no contexto de SOAD e alguns cenários de validação destes temas são propostos.
- [Teale & Jarves, 2004] propõem um método para identificação de componentes de software baseado em *clustering* em uma matriz CRUD feita de funções do negócio e entidades do domínio.

Nenhum destes métodos apresenta um procedimento detalhado para identificar, especificar, e modelar elementos arquiteturais apropriadamente (com métricas de desempenho aceitáveis) a partir de modelos de negócio.

A proposta baseia-se no conceito de Processo de Negócio Elementar (EBP) a fim de ter o menor nível de detalhes do modelo de processo de negócio que seja útil como uma visão conceitual

EBP é definido por [Mcsheffrey, 2001]: "Um processo realizado por uma pessoa em um lugar em um momento que adiciona valor e deixa os dados em estado consistente."

O modelo tem como entrada o modelo de processo de negócio no nível elementar e modelo de entidades da empresa, que descreve o modelo do domínio da empresa.

Consideram dois critérios para construção de modelos de negócio:

- Modelos de processos devem estar no menor nível necessário para a visão conceitual.
- A granularidade das entidades do negócio deve estar no nível em que cada entidade é criada em apenas em um processo elementar.

A proposta baseia-se na construção do modelo de serviços da empresa (ESM) a partir do modelo de negócio (EBM) da empresa em 4 passos:

1. Modelagem de processos de negócio
2. Identificação dos elementos do modelo de serviços
3. Categorização de serviços
4. Documentação do modelo de serviços da empresa.

A figura 1 apresenta os papéis responsáveis por executar cada passo da metodologia. Estes quatro passos são apresentados a seguir.

A **modelagem de negócio** (1) é feita por analistas de negócio e os modelos são elaborados segundo os critérios definidos anteriormente.

A **identificação dos elementos do modelo de serviços** (2) usa a técnica de análise de afinidade entre processos de negócio elementares (EBP) e entidades do negócio (BE) (EEAT – *Elementary business process and business Entity Affinity analysis Technique*).

Uma matriz EBP (linha)× BE (coluna) é criada, cada célula recebe um valor CRUD. Cada BE (coluna) só pode ter uma operação *create* e cada linha não deve ter nenhuma atividade (operação) conceitual. O objetivo é encontrar grupos de EBPs e BEs que compartilham operações *create* e *update*. O modelo é ajustado para colocar juntos processos de negócio e entidades do negócio com grande afinidade, em um método de clusterização que não é detalhado no artigo. O resultado é ter áreas exclusivas de EBPs e BEs formados ao redor de operações *create* e *update*. O objetivo é formar blocos não redundantes (Serviços da Empresa). Um exemplo é apresentado na Figura 18.

A especificação da estrutura e do comportamento de cada serviço pode ser especificada após a identificação dos serviços através da matriz:

- EBP localizados no agrupamento do serviço definem as operações do serviço, por exemplo, EBP1 a EBP8 definem as operações do serviço 1.

- Operações CRUD localizadas fora dos limites do serviço identificam canais de comunicação entre serviços. Por exemplo, o serviço 4 está relacionado com EBP15, a qual consulta BE2 (célula EBP15, EB2 marcada com R). Logo existe um canal de comunicação entre o serviço 4 e o serviço 1.
- Partições de serviço também podem ser realizadas na matriz (quadro com borda mais espessa).
- Processos de negócio são considerados consumidores dos serviços.
- Cada serviço tem um provedor que é analisado mais tarde no artefato de modelo de projeto. Por exemplo, serviço 2 tem 7 casos de uso e 4 entidades (BE2, BE3, BE4 e BE7). Este modelo também sofre agrupamento para representar componentes do serviço.

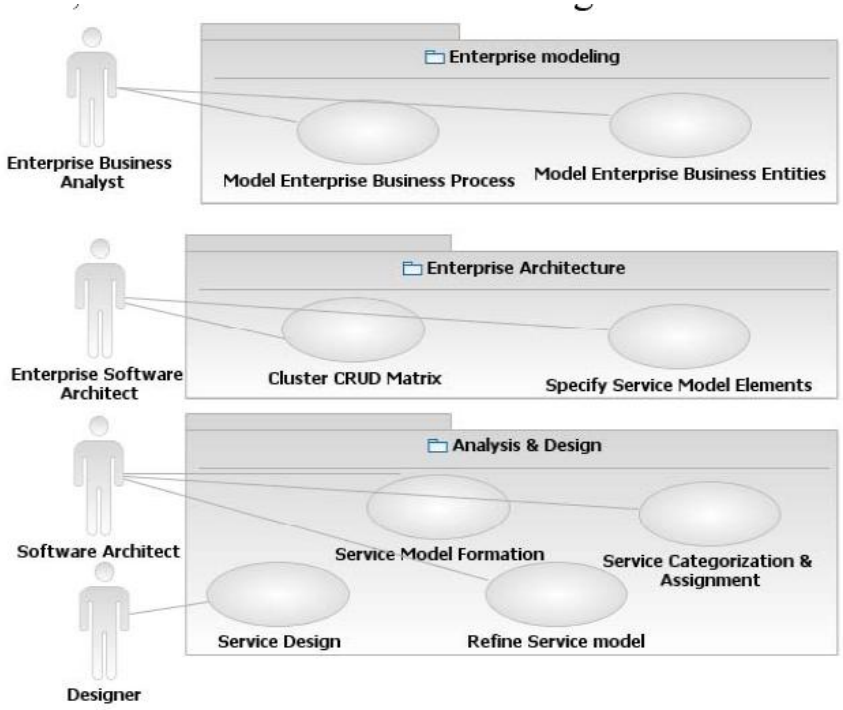


Figura 17 – Derivando ESM do EBMN

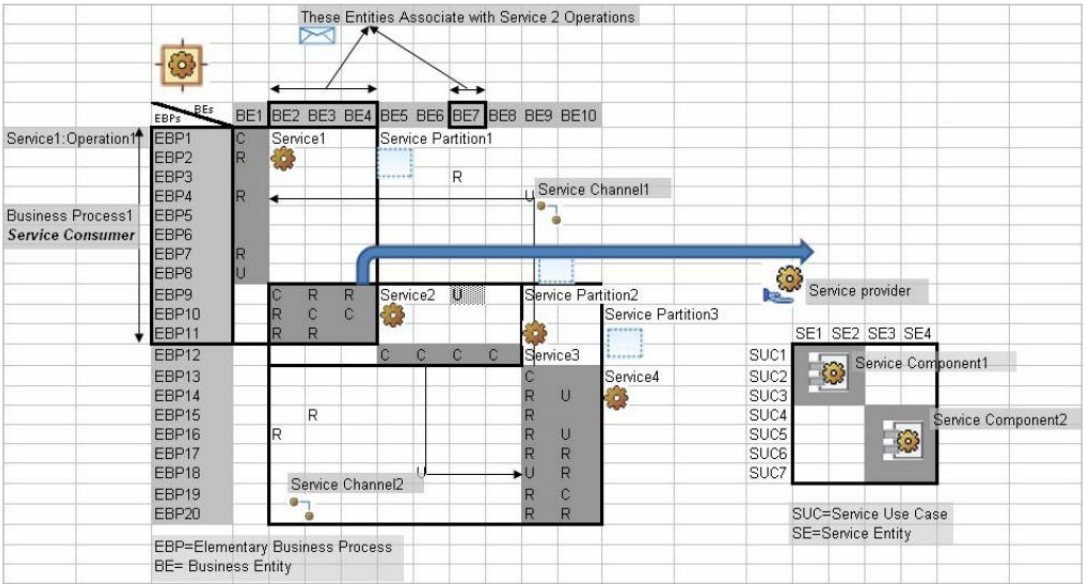


Figura 18 – Matriz de agrupamento CRUD

As atividades do passo 2 são:

- Construir matriz
- Comutar colunas
- Agrupar
- Indicar nomes dos serviços
- Indicar especificação do serviço
- Representar canais de colaboração de serviços
- Marcar consumidores do serviço
- Modelar provedores de serviço com componentes de serviço.

A **categorização de serviços** (3) corresponde a escolher se os serviços devem ser implementados em uma ferramenta BPMS ou como serviços externos à ferramenta. A maioria das arquiteturas de solução para SOA utilizam ferramentas de BPMS (*Business Process Management Systems*). Escolher a segmentação correta da lógica de negócio entre processos que são implantados em BPMS e serviços externos é muitas vezes desafiante e crucial. Uma regra é que serviços externos devem ser responsáveis por computação intensa e lógica complexa; enquanto que processos contém lógica que pode mudar em resposta a mudanças dos requisitos do negócio.

Portanto, alguns serviços são modelados em BPMS como uma parte do processo de negócio enquanto que outros são implementados como ferramentas para desenvolvimento de serviços.

A **documentação do serviço** (4) deve ser realizada um ferramentas de modelagem CASE. Neste trabalho eles usam o *profile* UML para modelagem de serviços [Johnson, 2005]. Este *profile* foi implementado no IBM Rational Software Architect (RSA) e IBM Rational Software Modeler (RSM), é tem sido usado com sucesso para desenvolver modelos de cenários complexos e para treinamento. O objetivo do *profile* é definir uma

linguagem comum para descrição de serviços através do ciclo de vida de serviços e prover visões (Figura 19.a) diferentes para cada *stakeholder*. Por exemplo:

- Visão de mensagens: contém modelos de classe representando as mensagens de entrada e de saída dos serviços (Figura 19.b).
- Visão de colaboração: exibe a colaboração de serviços existente nos processos de negócio subjacentes (Figura 19.c).
- Visão de serviços: contém definições, especificações, provedores, partições e dependências/associações entre serviços (Figura 19.d).

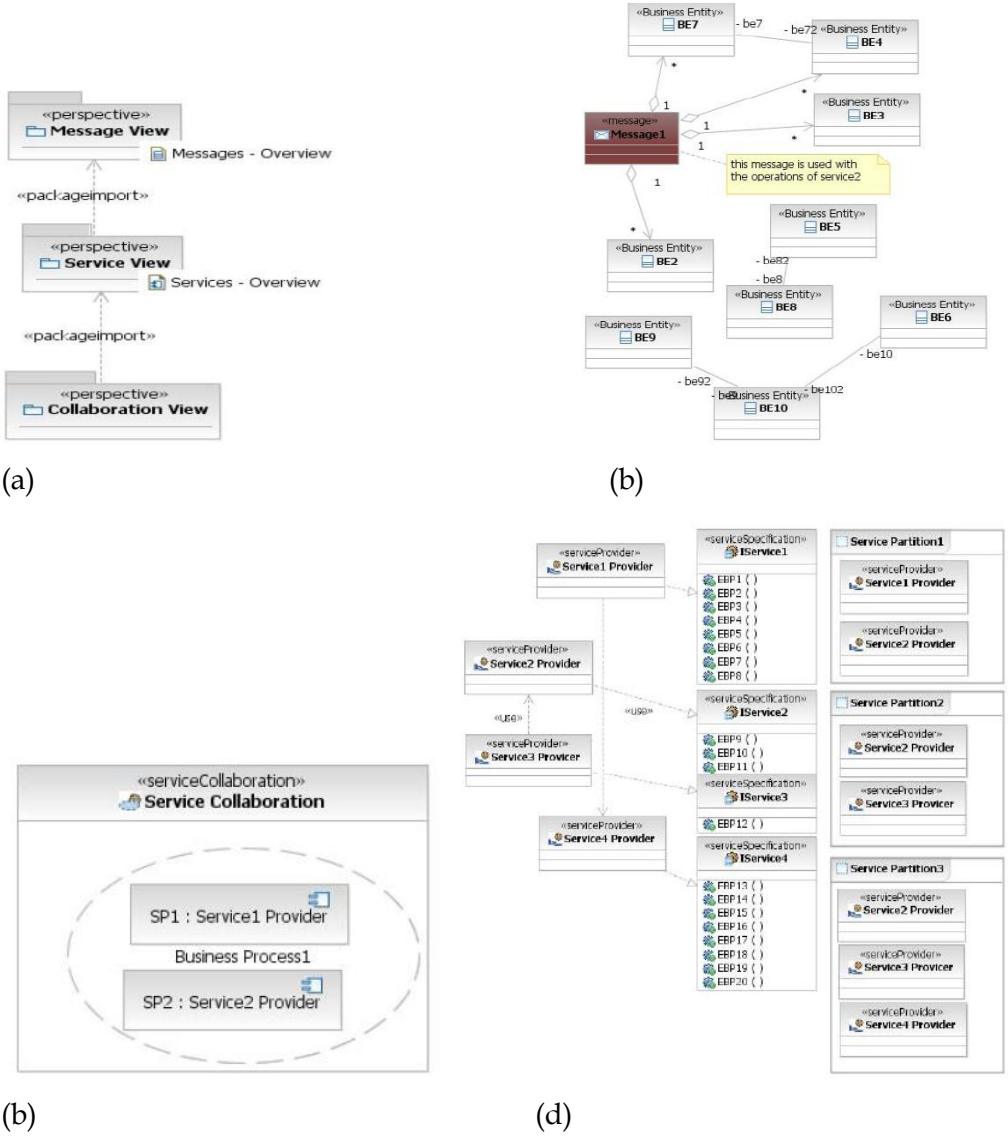


Figura 19 – Modelos de serviços

O autor apresenta um conjunto de métricas e seus respectivos fatores de influência a serem utilizadas para satisfazer critérios de desempenho, as quais são: reusabilidade, eficiência do reuso (contribuição do serviço para construir aplicações), manutenibilidade, granularidade, valor para o negócio, coesão e acoplamento. Como estas métricas de certa forma são mutuamente exclusivas, não é esperado satisfazer

todas elas. Logo, identificação de serviço é um problema de otimização com múltiplos objetivos.

O *framework* proposto por [Sol, 1990] foi utilizado para avaliar a proposta. Este *framework* define um conjunto de fatores essenciais que caracterizam um processo de desenvolvimento de sistemas de informação e os classifica em: forma de pensamento, forma de modelagem, forma de trabalho, forma de controle [Sol, 1990; Stojanović, 2005]. Como a proposta trata de análise e projeto de serviços, aspectos do *framework* de implementação foram desconsiderados. Foram então construídos questionários que foram submetidos a usuários avaliarem o método bem como para especialistas comparem os potenciais benefícios em relação a métodos tradicionais, principalmente RUP. Os questionários apresentavam questões, como apresentadas a seguir, e cada questão foi respondida com um peso variando de 1 (discorda fortemente) a 5 (concorda fortemente).

Portanto, o método de identificação e especificação de serviços foi executado e questões foram respondidas, em relação aos temas:

- Capacidades genéricas, por exemplo, "O método demonstrou a importância de um processo de identificação e especificação de serviços na construção de sistemas?"
- Capacidades específicas, por exemplo, "O conceito do serviço foi claramente definido?", "Os serviços identificados estavam viáveis em relação a métricas de desempenho?".

A.5.2 Análise da proposta

Este artigo propõe um novo processo adequado para identificação e especificação de serviços e de seus elementos arquiteturais a partir de modelos de negócio. A proposta busca satisfazer métricas de desempenho gerencial e técnica.

Uma nova técnica de agrupamento (*clustering*), chamada de processo de negócio elementar e técnica de análise de afinidade de entidade de negócio (EEAT), é apresentada para identificar elementos arquiteturais candidatos, com nível de granularidade correta, satisfazendo os princípios de baixo acoplamento, alta coesão e baixo custo de reuso.

Como positivo, o artigo apresenta conceitos, princípios, elementos de modelo e diretrizes para a construção de soluções orientadas a serviços a partir da modelagem de processos de negócio. Deve-se enfatizar o passo 2 (de identificação de serviços utilizando uma metodologia de agrupamento) e o passo 4 (de documentação dos serviços). O passo 1 corresponde a modelagem de processos de negócio e o passo 3 corresponde à definição se o serviço vai ser implementado como um artefato de um BPMS ou como um serviço construindo em uma arquitetura SOA.

Como ponto negativo está o fato do método basear-se em EBP (processos de negócio elementares) que é algo subjetivo para ser alcançado. Além disso, a proposta trata todos os serviços como sendo do mesmo tipo e identificados a partir de EB (Entidades do Negócio), não havendo separação entre serviços de negócio e serviços de dados, por exemplo.

O artigo propõe um novo método para identificação e especificação de elementos arquiteturais de uma arquitetura orientada a serviços. O método trata métricas de

desempenho de serviços. O método foi avaliado na prática, com serviços identificados e, em seguida, especialistas avaliaram o uso do método respondendo questionários em relação a capacidades genéricas e capacidades específicas. Foram obtidos bons resultados.

Como trabalhos futuros os autores propõem o desenvolvimento de um *framework* formal para suportar o método e construir um conjunto de ferramentas para apoiar o processo.

A.6 Second generation web services-oriented architecture in production in the finance industry

A.6.1 Análise da proposta

O artigo [Zimmermanm *et al.,* 2004b] apresenta um estudo de caso da implementação de serviços e as decisões tomadas nesta implementação. No entanto, por ser um artigo de 2004 muitos dos problemas já foram resolvidos. Por exemplo, SOAP já é um protocolo estabelecido, algumas ferramentas comprimem mensagens SOAP, etc. No entanto alguns pontos vale ressaltar.

As decisões da arquitetura foram baseadas nos requisitos que eles definiram:

- Melhorar a documentação dos serviços e o tempo de integração com aplicações, (WSDL foi utilizado),
- Minimizar o número de implementações de interfaces e *middlewares*, reduzir o volume de dados transferidos (utilizaram SOAP). Na ultima, escolheram entre um SOAP document/centric ou rpc/encoded. Contudo, o WS-I não recomenda mais o rpc/centric),
- Suportar um grande número de serviços (usaram um repositório próprio como se fosse um UDDI).

As seguintes boas práticas foram apresentadas:

- Seguir o princípio *design-by-contract* e utilizar ferramentas para geração de WSDL. O que já padrão atualmente.
- Selecionar os estilos de mensageria baseado nas características de interoperabilidade e suporte de ferramentas.
- Avaliar cuidadosamente qual estratégia para encontrar serviços é a melhor para o cada caso:
    - Usar UDDI na Web é problemático não por questões técnicas, mas organizacionais, uma vez que a web é um ambiente publico.
- Aplicar padrões pragmaticamente, de acordo com os interesses da organização.

A.7 Patterns: Service-Oriented Architecture and Web Services

A.7.1 Descrição da proposta

Este trabalho [Endrei, 2004] tem como objetivo introduzir e descrever padrões para comércio eletrônico (*e-business patterns*) desenvolvidos pela IBM, disponíveis em: http://www.ibm.com/developerworks/patterns/. Além disso, apresenta diversos conceitos de SOA e Serviços Web, indica algumas boas práticas na área e descreve tecnologias da IBM.

Os padrões para o comércio eletrônico da IBM representam um conjunto de arquiteturas de uso comprovado. O repositório de padrões, disponível em http://www.ibm.com/developerworks/patterns/, pode ser utilizado por empresas para facilitar o desenvolvimento de aplicações baseadas na web. Eles ajudam a compreender e analisar problemas complexos no negócio de uma organização e dividi-los em problemas menores, mais gerenciáveis e funções que podem então ser implementadas.

Neste *redbook*, é fornecida uma visão geral de como arquitetos podem trabalhar efetivamente com os padrões para e-business.

SOA tem implicações não apenas no nível da tecnologia, mas pelo ponto de interseção entre negócios e tecnologia. Ela ajuda a preencher essa lacuna, trazendo um maior foco para as empresas como descobrir e expor serviços. Estes serviços, muitas vezes, mas não necessariamente sempre, estão ao nível dos casos de uso de negócio.

Identificar o conjunto de serviços que uma empresa necessita para apoiar o seu negócio não é uma tarefa trivial. Analistas concordam que uma abordagem ou método é necessário para descobrir serviços alinhados com negócio, as suas dependências, bem como a granularidade de seus componentes.

A IBM propõe uma abordagem delineada por um conjunto de sete etapas que suportam a concepção dos componentes de base e arquiteturas orientadas para os serviços, baseada no uso dos padrões propostos. Passos para implementar uma abordagem SOA:

1. Decomposição do domínio do negócio (identificação dos processos e sua decomposição – casos de uso de negócio)
2. Criação de um modelo objetivo-serviço (especificação de objetivos e sub-objetivos relacionados a cada processo e conseqüente definição de serviços candidatos para cada sub-objetivo)
3. Alocação de serviços (alocação dos serviços a cada sub-objetivo)
4. Especificação de componentes
5. Uso dos padrões para estruturar componentes e serviços (estudo de cada situação e aplicação dos padrões nos vários níveis)
6. Mapeamento da realização da tecnologia

Os padrões para e-business são dispostos em um conjunto de camadas estruturadas que podem ser exploradas por qualquer metodologia de desenvolvimento:

- *Business patterns*: identificam a interação entre usuários, empresas e dados.

- *Integration patterns*: integram vários *business patterns* em um conjunto, quando uma solução não pode ser fornecida com base em um único *business pattern*.
- *Composition patterns*: representam combinações comuns que ocorrem entre *business patterns* e *integration patterns*.
- *Application patterns*: provêem um esquema conceitual descrevendo como componentes de dados da aplicação interagem dentro de um *business pattern* ou *integration pattern*.
- *Runtime patterns*: definem a estrutura *middleware* lógica que apóia um *application pattern*. Descrevem os principais nós, seus papéis, e as interfaces entre esses nós.

O trabalho também estabelece um conjunto de boas práticas em projetos de serviços (Boas Práticas em Projetos: http://www.ibm.com/developerworks/webservices/), resumidos na seção 4.1.7 .

A.7.2 Análise da proposta

O trabalho é bastante completo, descreve em detalhes e de forma bastante clara quase todos os conceitos e tecnologias relacionados à SOA e *Web Services*. Pode ser usado como referência para entendimento e aprendizagem. Além disso, apresenta passos para uma metodologia voltada para SOA que incorpora o uso dos padrões propostos e exemplifica quando e como utilizá-los.

O ponto negativo (para o projeto) é que não detalha ou apresenta heurísticas específicas para a questão da granularidade de serviços. No entanto, aponta uma lista de boas práticas que poderia ser estendida, ou mais bem trabalhada, para ajudar a gerar as heurísticas desejadas.

As boas práticas de projeto sugeridas no trabalho podem ser usadas como base para definição das heurísticas de projeto de serviços. Feita uma análise pela equipe de desenvolvimento, os padrões descritos podem ser adotados como boa prática também.

Boas práticas em outras etapas do processo também são especificadas e podem ser úteis nas próximas fases do projeto.

A.8 An Automated Method for Service Specification

A.8.1 Descrição da proposta

Jamshidi *et al.* [2008] apresentam que pesquisa sobre padrões, produtos, tecnologias e computação orientadas a serviços tem sido abundante, mas poucas experiências práticas e teóricas têm sido relatadas em análise e projeto de serviços. Ainda assim, os poucos trabalhos publicados sofrem de sérias deficiências [Zimmermann *et al.,* 2004a] e [Amsden, 2007]. Eles não são aplicados na prática nos níveis da empresa, os serviços identificados não satisfazem os objetivos da identificação tais como métricas de desempenho técnico e de gerenciamento, e os processos propostos são ambíguos e sem rastreabilidade. Além disso, os métodos propostos não são automatizados, focando em processos baseados em decisões manuais suscetíveis a erro e, nos casos mais favoráveis, são introduzidos processos de

especificação semi-automática através de padrões de projeto reutilizáveis [Jamshidi *et al.,* 2009]. O que é necessário é um modelo mediado para formalizar e unificar modelos de negócio e produzir o nível de abstração adequado a fim de que decisões arquiteturais possam ser tomadas automaticamente [Jamshidi *et al.,* 2008].

Definir e aplicar transformação de modelo é uma técnica crítica para automatizar o processo de desenvolvimento [Brown *et al.,* 2005]. Além disso, adotar métricas técnicas de um estilo arquitetural é uma tática complementar para ter uma solução eficiente e com qualidade [Pressmann, 2004]. Portanto, através da introdução de um modelo de transformações, o qual utiliza métricas técnicas para automatizar o ciclo de vida de modelagem orientada a serviços, nós podemos alcançar estes objetivos.

O método proposto é apresentado na Figura 20 e tem como entrada uma matriz CRUD gerada pela aplicação do método ASIM (*Automatic Service Identification Method*) e uma matriz de processos de negócio BP e EBP. Esta matriz é usada durante o projeto de interação de serviços para determinar os elementos comportamentais do modelo de serviço.

A etapa "Estruturar arquitetura do serviço"(*Structure service architecture*) possui as seguintes sub-etapas:

- Modelar mensagens: corresponde à elaboração de um diagrama de mensagens de entrada e saída de serviços;
- Modelar especificação dos serviços: corresponde a modelar a interface dos serviços, com suas operações e mensagens de entrada e saída. As especificações dos serviços são geradas a partir da matriz CRUD:
    - EBPs que lidam com pelo menos uma BE são interpretados como operações do serviço,
    - Mensagens de entrada e saída das operações são extraídas de BEs que o EBP manipula:
        - Criação de uma BE corresponde a uma mensagem de saída;
        - Leitura de uma BE corresponde a uma mensagem de entrada;
        - Atualização de uma BE corresponde a uma mensagem de entrada e uma de saída;
        - Remoção de uma BE corresponde a uma mensagem de entrada.
- Modelar provedores dos serviços: corresponde a modelar provedores de serviço, suas partições e seus *gateways*. O agrupamento de serviços para expô-los em um único provedor é feito de acordo com a intensidade do relacionamento semântico entre serviços, o qual é definido pela intensidade de operações "R", "U" e "D". *Gateways* correspondem a serviços fachada (*façades*), expostos pelo provedor de serviço, que interagem com serviços fora da partição, e os quais têm alta intensidade de relacionamento entre provedores de serviço. Este serviço provavelmente será um serviço composto ou participará de pelo menos uma orquestração de serviço.

A etapa "Modelar o comportamento dos serviços" (*Model service behavior*) é dividida em duas sub-etapas:

- Modelar a colaboração de serviços: corresponde à elaboração de um diagrama de colaboração contendo elementos do modelo do provedor de serviços que participam da colaboração, serviços que eles provêem, e canais de serviço que formam a relação entre serviços e provedores de serviços. Canais de serviços são identificados através das operações "R", "U" e "D"que estão fora do cluster do serviço, na matriz CRUD.
- Modelar a interação entre serviços: corresponde à composição de serviços ou orquestração de serviços que são modelados em um diagrama de interação através de um diagrama de sequência ou diagrama de atividades.

Na etapa "Criar o modelo de projeto inicial" são tratados aspectos da realização de serviços. Neste passo, componentes dos serviços são identificados. Componentes de serviço são definidos segundo a proposta de [Johnston & Smith, 2005] o qual identifica um componente de serviço para provedores de serviço, os quais nenhum dos serviços é promovido. Para provedores de serviço que pelo menos um dos seus serviços é promovido, serviços não promovidos são considerados para serem realizados como um componente de serviço e cada serviço promovido é considerado como um componente de serviço distinto.

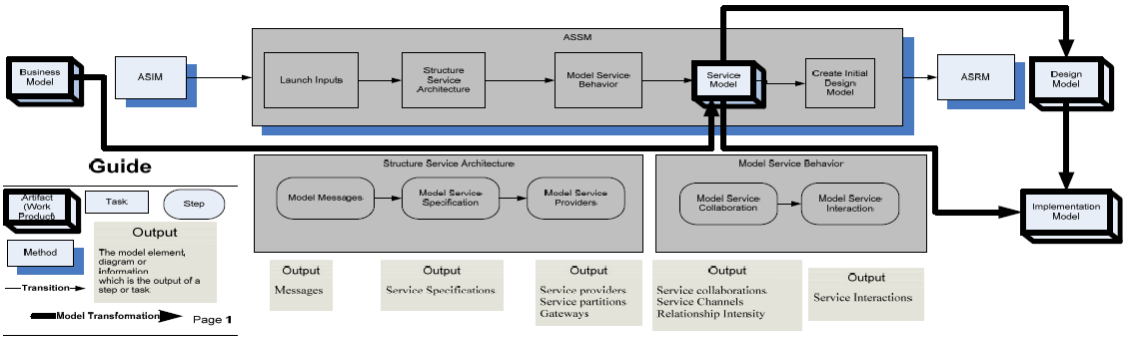


Figura 20 – Passos do método proposto

A.8.2 Análise da proposta

Jamshidi *et al.* [2009] propõem uma maneira automatizada para geração de artefatos para a especificação de serviços. No entanto, a proposta é apresentada de forma genérica e sem um exemplo prático, o que dificulta o entendimento e instanciação da proposta na prática (o que não é também apresentado no artigo).

A.9 Deriving Software Services from Business Process of Representative Customer organizations

A.9.1 Descrição da proposta

Adam *et al.* [2008] tratam o desafio de identificar serviços no nível adequado de abstração e com um conjunto de funções correto. Conjunto de funções refere-se à quantidade de operações diferentes providas pelo serviço, enquanto que nível de abstração descreve se uma operação realiza uma funcionalidade mais orientada ao negócio ou uma funcionalidade mais técnica. O artigo apresenta um método para

derivar sistematicamente serviços (*web services*), baseado nos processos de negócio de um conjunto representativo de clientes de uma organização. Dessa forma, o artigo tem o objetivo de identificar serviços entre organizações, e não dentro da própria organização. No contexto do EP, onde cada unidade organizacional é grande o suficiente e com quantidade de processos suficiente para ser considerada como uma organização, o conteúdo do artigo torna-se relevante para o projeto.

Os autores apontam que o método pode ser utilizado por empresas interessadas em reestruturar seus sistemas monolíticos em serviços menos acoplados, por exemplo, considerando processos AS-IS. O método também pode ser utilizado por organizações que buscam o desenvolvimento de serviços inovadores, quando processos TO-BE devem ser considerados.

Adam *et al.* [2008] propõem os seguintes tipos de serviços:

- Serviço de negócio: são serviços providos por uma organização para seus clientes ou por uma unidade de negócio a outras unidades de negócio. Por exemplo, uma companhia de seguros oferece o serviço "seguro". Um serviço de negócio engloba um conjunto de atividades do negócio.
- Serviços de aplicação: são serviços que realizam funções (atômicas) do negócio. Segundo Adam *et al.* [2008], um conjunto de funções atômicas do negócio realizam uma atividade elementar Uma atividade elementar corresponde a um nível de refinamento das atividades de um processo de negócio no nível de que pode ser executada por um papel que pode ser um ser humano ou serviços de aplicação. Logo, cada passo de uma atividade elementar é traduzido em um serviço de aplicação.
- Serviços de processo: são serviços que provêem sequencias de interações e não apenas funcionalidades atômicas, ou seja, são serviços que orquestram outros serviços como definido também em [Josuttis, 2007].

O artigo focaliza em serviços de aplicação, os quais são responsáveis por automatizar passos de uma atividade elementar e são "produtos" de software, por exemplo, *web services*, que provêem suporte (com valor) para os trabalhos diários de organizações clientes. Conseqüentemente, um serviço de aplicação possui as seguintes características:

- Bem empacotados: suas operações formam um conjunto apropriado para todos os clientes possíveis;
- Coeso: as funções providas têm um grande relacionamento;
- Relacionados ao negócio: as operações do serviço apóiam atividades do negócio diretamente e não se tratando de apenas funcionalidades de TI ;
- Independência de contexto: operações podem ser utilizadas independentes do contexto de onde são invocadas;
- Funções de software: as operações do serviço não requerem internamente intervenção humana, e simplesmente transformam entradas (do usuário) em saídas.

Muitas das características de serviços apresentadas são melhor tratadas na fase de análise e projeto. O artigo apresenta algumas propostas para tratá-las. No entanto, o artigo não trata como os serviços são estruturados ou realizados internamente.

O artigo propõe um método para definir o nível de abstração e funcional mais adequado para um serviço. É proposto que o os processos de negócio sejam especificados considerando requisitos não funcionais no nível de atividades elementares a fim de derivar diretamente um conjunto de funcionalidades do serviço. Os autores definem atividade elementar como sendo uma atividade descrita em mais detalhes, por exemplo, a descrição da atividade inclui como um papel deve interagir com um sistema. A proposta é que uma atividade elementar seja descrita usando um *template* de caso de uso mínimo [Cockburn, 2001] a fim de explicitar todos os passos de interação alternados executados para realizá-la.

Cada passo do detalhamento da atividade elementar é transformado em um serviço. Inicialmente, os parâmetros de entrada e saída do serviço são obtidos a partir de uma das atividades em que o serviço está inserido.

O serviço pode ser extraído a partir de atividades distintas, em contextos distintos, para executar a mesma funcionalidade. Por outro lado, os serviços devem atender ao princípio de serem independentes de contexto. [Adam et al. 2008] propõem que serviços sinônimos sejam consolidados e que a assinatura mais genérica do conjunto de serviços seja considerada. Além disso, como as atividades podem requerer a execução do serviço com diferentes requisitos funcionais, deve ser considerado para o serviço o requisito funcional mais restritivo.

Como resultado, têm-se um conjunto disjunto de operações de serviço necessárias para suportar todos os processos de negócio específicos de clientes.

No passo seguinte da metodologia, funcionalidades do serviço são agrupadas. Tipicamente, funcionalidades do serviço são agrupadas baseando-se no tipo de funcionalidades ou baseando-se na estrutura de dados que elas operam. Os autores propõem o uso de ambas as abordagens. Por exemplo, todas as funções atualizando ou modificando uma entidade (ou seja, operações de CRUD) devem ser agrupadas em um bloco de funções relacionadas à entidade. Além disso, devem ser considerados os objetivos do software a ser desenvolvido de modo que funcionalidades e blocos de funcionalidades que não se pretende desenvolver, devem ser descartados.

A proposta considera que os serviços são identificados a partir dos modelos de processos de negócio dos clientes da organização que está desenvolvendo os serviços.

Os autores propõem um agrupamento de funcionalidades de acordo com os clientes que usam o serviço. Inicialmente devem ser removidos os *outliers* que são os exemplos de uso que diferem de todos os outros. Os casos em que uma funcionalidade é muito utilizada por determinados clientes e não é utilizada por outros clientes devem ser analisadas, para verificar se realmente o cliente não poderia usufruir da funcionalidade. Quando um cliente usa apenas poucas funcionalidades de um grupo, deve ser analisado se a funcionalidade é realmente necessária. Se não houver argumento suficiente para justificar o uso das funcionalidades pelo cliente, a marcação deve ser removida.

O agrupamento de serviços inicialmente é realizado baseado na experiência de um especialista. Como isto pode não refletir a necessidade dos clientes, agrupando em um serviço funcionalidades que não interessam ao cliente, os autores propõe o uso da correlação φ para validar a demanda de duas funcionalidades estarem no mesmo serviço. Devem ser combinados em um serviço apenas as funcionalidades que possuem alta correlação. Os autores ainda propõem uma simplificação computacional

para o método, agrupando em um mesmo serviço todas as funcionalidades que têm o mesmo uso. Eles advertem ainda que aplicar o método desta forma não garante que o agrupamento será o mais adequado.

Na etapa de especificação do serviço, os autores propõem definir nomes mais genéricos para as funções e considerar como requisito não funcional aquele que é mais restritivo.

A proposta foi testada na prática em dois projetos reais de clientes do Instituto Frauhofer em que os autores trabalhavam. Um dos projetos era de um cliente da indústria com interesse em estender seu portfólio de serviços, enquanto que outro estudo de caso foi realizado para melhorar TI para organizações públicas. A partir destas aplicações, os autores listaram as seguintes lições aprendidas:

- Os processos a serem considerados para derivação dos serviços não devem ser escolhidos "ad-hoc", sendo necessário uma forma sistemática para definir escopo.
- O envolvimento de todos os *stakeholders* dos processos é crucial, não devendo ser considerados apenas o conjunto de *stakeholders* capaz de explicar melhor as características do negócio.
- Foi observado que processos as-is poderiam melhorados utilizando técnicas de melhoria de processos de negócio
- Raramente é necessário elicitar todos os aspectos do negócio, porque normalmente, nem todos os serviços estão no escopo do desenvolvimento de software. Dessa forma, a análise de processo de negócio é deve ser uma fase conduzida antes de iniciar o processo de identificação de serviços.
- Na fase de identificação de serviços, a definição da assinatura dos serviços não deve ser muito detalhada, pois esta será realmente definida quando detalhes técnicos forem considerados.
- Agrupar funções de serviços baseado nas estruturas que eles operam é uma abordagem que produz bons resultados. No entanto, agrupar funcionalidades que executam processamento puro ainda é um desafio a ser pesquisado.
- Observaram que o agrupamento de operações pelo tipo de funcionalidade usando a técnica de agrupamento proposta gerou agrupamentos muito instáveis. Executando o agrupamento com menos de 10 clientes não produziu resultados confiáveis.

A.9.2 Análise da proposta

Muitas das características de serviços apresentadas, tais como, empacotamento, coesão, são melhor tratadas na fase de análise e projeto. O artigo apresenta algumas propostas para tratá-las. No entanto, o artigo não trata como os serviços são estruturados ou realizados internamente.

É proposto que o os processos de negócio sejam especificados considerando requisitos não funcionais no nível de atividades elementares a fim de derivar diretamente um conjunto de funcionalidades do serviço. Nos modelos de processos elaborados na GDIEP, não estão definidos requisitos não funcionais.

Os autores definem atividade elementar como sendo uma atividade descrita em mais detalhes, por exemplo, a descrição da atividade inclui como um papel deve interagir com um sistema. A proposta é que uma atividade elementar seja descrita usando um *template* de caso de uso mínimo [Cockburn, 2001] a fim de explicitar todos os passos de interação alternados executados para realizá-la. Os modelos de processos elaborados atualmente na GDIEP não chegam neste nível de abstração. Este detalhamento é muitas das vezes não é o objetivo da modelagem de processos de negócio. Logo, ou o método por nós proposto deve considerar esta característica, ou devemos explicitar a necessidade de se detalhar os processos que serão disponibilizados como serviço.

Inicialmente, os parâmetros de entrada e saída do serviço são obtidos a partir de uma das atividades em que o serviço está inserido. Na nossa proposta, como consideramos requisitos e regras de negócio, temos um nível de granularidade tal que conseguimos ter os parâmetros de entrada e saída do serviço mais explícitos e mais genéricos. O mesmo não ocorre no caso de serviços identificados a partir de AND, XOR etc.

Serviços semelhantes devem ser consolidados em um serviço. A assinatura do serviço deve ser definida de acordo com a função mais genérica. Se as funções consideram diferentes requisitos não funcionais, então deve ser considerado o requisito funcional mais restritivo.

Todas as funções atualizando ou modificando uma entidade (ou seja, operações de CRUD) devem ser agrupadas em um bloco de funções relacionadas à entidade.

Como pontos positivos:

- A proposta trata a identificação de serviços a partir de modelos de processos de diferentes clientes da organização, o que é semelhante a nossa contexto, onde temos unidades de negócio diferentes com modelos de processos diferentes que usam o serviço.
- A granularidade dos serviços está no nível que será utilizada pelos clientes.

Como pontos negativos:

- Não é garantido que a execução do método agrupe serviços da melhor forma possível.
- A granularidade dos serviços considerada é sempre fina.

A.10 Identification and Analysis of Business and Software Services – A Consolidated Approach

A.10.1 Descrição da proposta

O objetivo do artigo [Kohlborn *et al.*, 2009] é apresentar um estudo e análise feitos sobre diversos métodos para identificação de serviços, verificando seus pontos fortes e fracos (com base em critérios definidos: Conceito de SOA, Estratégia de implementação de SOA, Cobertura do ciclo de vida, Grau de prescrição, Acessibilidade e validade). A partir do resultado obtido, propõe um novo método que contemple todos os requisitos considerados.

Os autores definem os seguintes conceitos:

- *Business service* (serviço de negócio): conjunto específico de ações executadas por uma organização, podendo estes serem definidos em diferentes níveis de granularidade. Os tipos de serviços de negócio devem ser classificados de acordo com a sua importância estratégica para a organização e seus gestores (por exemplo, grupos de grupo, serviços de unidades organizacionais, serviços de equipes, etc.)
- *Software service* (serviço de software): parte de um sistema o qual pode ser consumido separadamente por diversas entidades, ou seja, provê suporte para a execução de um serviço de negócio. Os tipos de serviços de software são tipicamente: serviços atômicos e serviços compostos.
    - Serviços atômicos podem ser: serviços utilitários (não estão relacionados à lógica do negócio, sendo processos transversais ao mesmo), serviços de tarefas (relacionados com uma tarefa ou processo de negócio) e serviços de entidades (representam um serviço centrado em negócio com fronteira que perpassa uma ou mais entidades relacionadas ao negócio).
    - Serviços compostos podem ser divididos em duas classes: serviços de agregação de lógica e serviços de agregação de dados.

Os autores levantam as seguintes questões: (1) Quais são os conceitos relevantes por trás dos métodos e abordagens para identificação e análise de serviços em uma organização? (2) Se existem deficiências nestas propostas. Como estes conceitos podem ser consolidados, melhorados e estendidos para prover um novo método?

O método para identificação de serviços proposto está estruturado da seguinte forma:

(1) Derivação de serviços de negócio
    - Fase de preparação
    - Fase de identificação
    - Fase de detalhamento
    - Fase de priorização

A Figura 21 apresenta o resultado de detalhamento do processo exemplificado no artigo

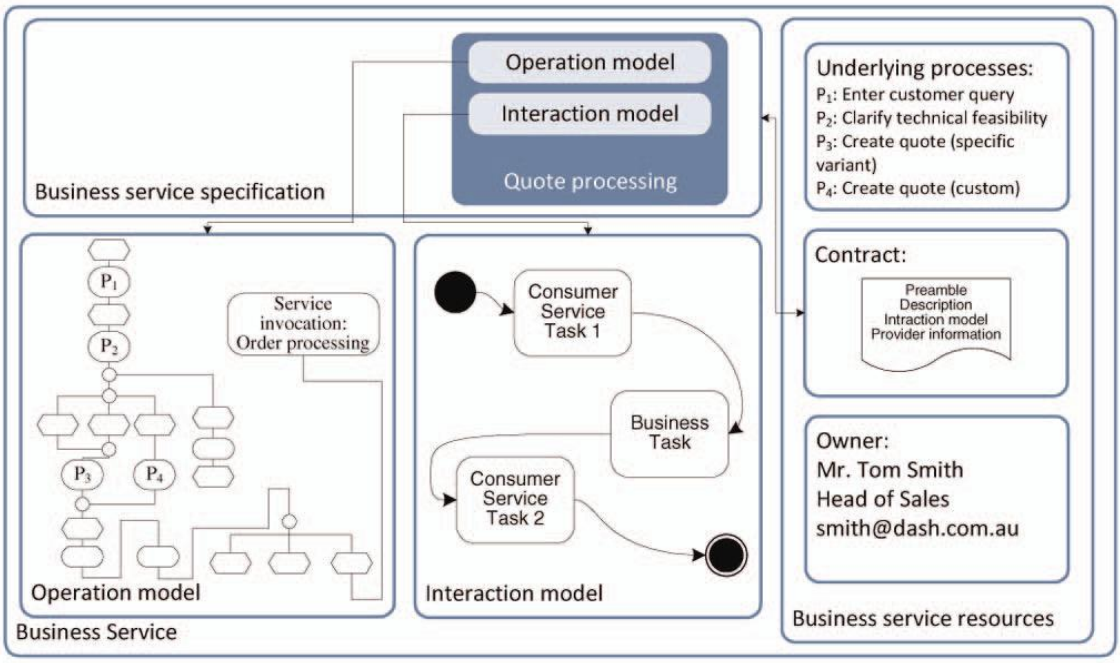


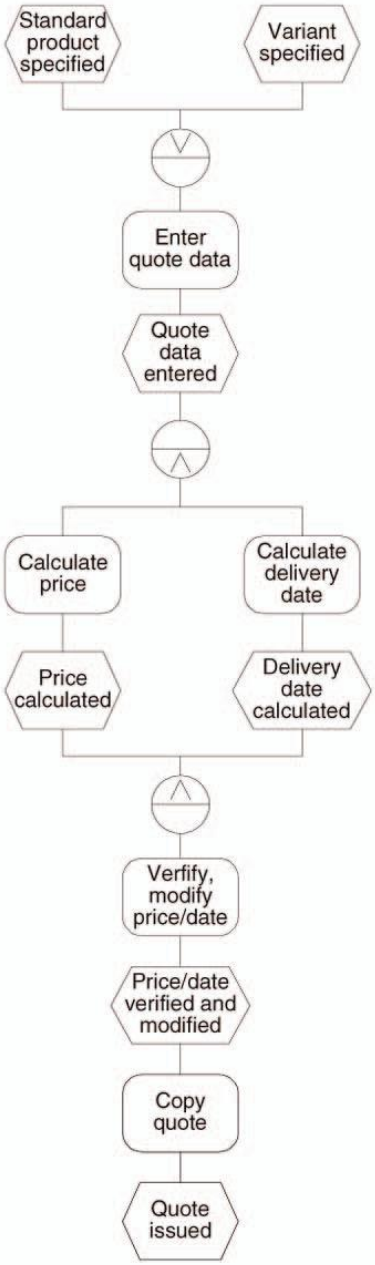
Figura 21 - Detalhamento do serviço de negócio Cotação

(2) Derivação de serviços de software

- Fase de preparação
  - De forma a prover base para a identificação de serviços de software o respectivo processo deve ser decomposto e refinado com informações sobre análise da aplicação. A saída desta fase é um conjunto de processos identificados como adequados para serem serviços e decompostos em passos granulares.
- Fase de identificação: as seguintes atividades são realizadas:
  - Identificar entidades correspondentes: usar os modelos de processos para identificar entidades do negócio manipuladas e operações usadas sobre estes objetos. Modelos de entidade-relacionamento, modelos de classes podem ser gerados para apontar serviços de entidade preliminares.
  - Analisar visibilidade e *takeover* (uso) dos passos do processo: analisar as funções do processo com base em sua visibilidade e potencial de interação com *stakeholders* para identificar potenciais agrupamentos de funcionalidades que devem ser expostas para os *stakeholders* através de serviços.
  - Identificar operações de serviços potenciais: cada passo do processo pode ser considerado a princípio uma operação.
  - Extrair a lógica do processo: extrair regras de negócio a partir do processo (atividades e fluxo)
  - Definir contextos lógicos: aplicar o princípio do acoplamento para identificar dependências seqüenciais entre operações.

- Definir composições: desenvolver cenários de uso dos serviços para apoiar a decisão de composição.

A Figura 22 apresenta o resultado da Análise do processo Criar cotação.



Function to be evaluated	Inter-organisational unit / IT System	Line of interaction / visibility	Group of stakeholders: Logistics Company	Supplier	Customer
Enter quote data (semi-automatic)	Sales; Quotation System	**Takeover**			X_1
		Visibility			X_2
		X_1:Data based on specification; X_2: Customer must only see own data			
Calculate price (automatic)	Quotation System	**Takeover**			X_3
		Visibility			X_2
		X_2 : Customer must only see own data; X_3: Only customer-specific calculation			
Calculate delivery date (automatic)	Quotation System	**Takeover**			X_3
		Visibility			X_2
		X_2 : Customer must only see own data; X_3: Only customer-specific calculation			
Verify, modify price/date (semi-automatic)	Sales; Quotation System	**Takeover**			
		Visibility			
Copy quote (automatic)	Quotation System	**Takeover**			
		Visibility			

Figura 22 – Análise do processo Criar cotação

- Fase de detalhamento: verificar serviços existentes que se sobreponham aos identificados e decidir por extensões, adaptações, etc. Para cada serviço identificado e verificado, detalhes específicos devem ser documentados - identificador, descrição, tipo, parâmetros de entrada e saída, consumidores (Tabela 16 e Tabela 17).

Tabela 16 - Detalhamento de Serviços de Tarefa, Entidade e Utilitários

Atomic services	Operation	Input Parameter	Output Parameter	Service Consumer
Quote (Entity)	create()	quote data [payment and delivery conditions]	quoteID	CU (customer)
	update()	quote data [payment and delivery conditions (delta)]	notification	
	read()	quoteID	quote data	CU
	delete()	quoteID	notification	
Price (Task)	calculatePrice()	materialID, values	price	CU
	modifyPrice()	quoteID, new Price	notification	
Delivery date (Task)	calculateDeliveryDate()	quoteID, values	delivery date	CU
	modifyDeliveryDate()	quoteID, new delivery date	notification	
Copy (Utility)	copy()	data	notification	

Tabela 17 - Detalhamento de Serviços de Processos

Process Service	Service Consumer	Function	Service	Operation
Enter quote		enter quote data	Quote	create()
		calculate price	Price	calculatePrice()
		calculate delivery date	Delivery date	calculateDeliveryDate()
		modify price, delivery date	Price	modifyPrice()
			Delivery date	modifyDeliveryDate()
		copy	Copy	copy()
Calculate quote	CU	calculate price	Price	calculatePrice()
		calculate delivery date	Delivery date	calculateDeliveryDate()
		enter quote data	Quote	create()

A.10.2 Análise da proposta

O método proposto trata de várias questões relevantes para identificação de serviços, reutilizando propostas da literatura. A proposta final parece ser um bom guia, no entanto, não apresenta um estudo de caso, apenas um exemplo de uso.