CLÁUDIO BIANCHI

MÉTODO PARA ANÁLISE, ESPECIFICAÇÃO, DESENVOLVIMENTO E GOVERNANÇA DE UMA SOA

Texto apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de MBA em Tecnologia da Informação

Área de Concentração:
Sistemas Digitais

Orientador:
Prof. Dr. Júlio Arakaki

São Paulo
2007

FICHA CATALOGRÁFICA

Bianchi, Cláudio

Método para análise, especificação, desenvolvimento e governança de uma SOA / C. Bianchi. -- São Paulo, 2007.
p.

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

1.Sistemas de informação 2.Sistemas distribuídos I.Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II.t.

AGRADECIMENTOS

Ao colega e orientador Prof. Dr. Júlio Arakaki pelas diretrizes seguras e permanente incentivo.

Aos colegas da turma do MBA em Tecnologia da Informação.

A todos que, direta ou indiretamente, colaboraram na execução deste trabalho.

RESUMO

Diversos estudos apontam que arquiteturas orientadas a serviço (SOA), dentro de alguns anos, terão grande influência sobre o desenvolvimento de novos sistemas. O Instituto Gartner (2006) estima que até 2008, mais do que 75 por cento das organizações usarão SOA no desenvolvimento e integração de aplicações e processos. O objetivo de uma SOA é permitir às empresas realizar negócio e ter vantagens tecnológicas por meio da combinação de inovação de processos, governança eficaz e estratégia de tecnologia, as quais giram em torno da definição e reutilização de serviços. Um dos benefícios mais importantes que SOA fornece é a melhora da produtividade e agilidade tanto para o negócio quanto para TI, e conseqüentemente, a redução de custos no desenvolvimento e manutenção dos sistemas envolvidos. Mas para desenvolver e implementar uma SOA, é necessário a utilização de um método abrangente de análise, especificação, desenvolvimento e governança que permita aos interessados tirar o melhor proveito deste novo tipo de abordagem, diminuindo os riscos inerentes a um projeto deste porte. O desenvolvimento deste método abrangente é o foco desta monografia. É proposto o método SOA-v360 que busca dar uma visão de 360º de todas as fases do ciclo de vida SOA.

ABSTRACT

Many researches point that service-oriented architecture (SOA), by the end of some years, will have a great influence about how the new systems are developed. The Gartner Institute (2006) estimates that by 2008, more than 75 percent of enterprises will use SOA on development and integration of aplications and processes. The goal of a SOA is to enable organizations to realize business and technology advantages through a combination of process inovation, effective governance, and a technology strategy that revolves around the definition and re-use of services. The key benefit that SOA provides is to improve productivity and agility for both Business and TI and consequently, the costs reduction in development and maintenance of the systems involved. But for SOA development and deployment, it is necessary a comprehensive method of analysis, design, development, implementation and governance that allow interested people take advantage of this new kind of approach, decreasing the inherent risks to a project like that. The development of this comprehensive model is the focus of this academic work. It is proposed the SOA-v360 method, which is proposed a 360º view of all SOA life cycle phases.

SUMÁRIO

LISTA DE FIGURAS E TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BPM	*Business Process Management* , ou Gerenciamento de Processos de Negócios
BPEL	*Business Process Execution Language* , ou Linguagem de Execução de Processos de Negócio
EAI	*Enterprise Application Integration,* ou Integração de Aplicações Corporativas
ESB	*Enterprise Service Bus*, ou Barramento Corporativo de Serviços
ROI	*Return on Investment*, ou Retorno sobre Investimento
SOA	*Service-Oriented Architecture,* ou Arquitetura Orientada a Serviço
SOA-v360	Método de desenvolvimento e governança SOA com visão 360°
SOAP	*Simple Object Access Protocol*
TI	Tecnologia da Informação
UDDI	*Universal Description, Discovery, and Integration*
WSDL	*Web Services Description Language*, ou Linguagem de Descrição de Serviços Web
XML	*Extensible Markup Language*, ou Linguagem Extensível de Formatação

1. INTRODUÇÃO

1.1. Objetivos

O principal objetivo deste trabalho é apresentar um método que auxilie na análise, especificação e desenvolvimento de arquiteturas corporativas de TI baseadas em serviços , mais conhecida como SOA. Para isto, se faz necessária a utilização de um método integrado que englobe as várias perspectivas de uma arquitetura corporativa, tais como a perspectiva do negócio, a perspectiva do planejamento e da governança corporativa, a perspectiva do desenvolvimento das aplicações e a perspectiva das operações.

Em cada uma das fases da perspectiva desenvolvimento, devem ser propostas as seguintes atividades:

- Como definir o escopo da análise orientada a serviço;
- Como identificar e modelar os serviços;
- A importância e como definir os requisitos não-funcionais;
- A especificação da tecnologia e escolha da plataforma de desenvolvimento da arquitetura orientada a serviços, bem como realizar a orquestração dos serviços com os processos de negócio.

Mas antes mesmo de iniciar as fases mencionadas de análise, especificação e desenvolvimento, é necessário o planejamento e a criação das condições necessárias para que a implementação de uma iniciativa deste porte seja bem sucedida. Assim, outro objetivo é apresentar um método de planejamento e governança de uma arquitetura orientada a serviços, o qual deverá abordar os seguintes aspectos:

- A importância de estabelecer uma visão SOA, na qual serão estabelecidos os limites e os objetivos que a implementação pretende alcançar;
- Como criar indicadores SOA com o Balanced Scorecard;
- A importância de calcular e quais os critérios que podem ser utilizados no cálculo do retorno do investimento de uma SOA;

- Como gerenciar os riscos na implementação de uma arquitetura SOA.

E antes de qualquer um dos pontos mencionados anteriormente, está a modelagem do negócio, onde se objetiva apresentar a importância de que os planos da arquitetura corporativa estejam alinhados com os objetivos, metas e estratégias corporativas, bem como definir os requisitos do negócio e modelar os processos de negócio.

1.2. Justificativa e Motivações

Reduzir custos tem sido a obsessão de todos os executivos que trabalham com processos administrativos desde o início dos tempos. Esta máxima de reduzir custos de todas as maneiras se aplica às áreas que não estão diretamente ligadas aos processos de linha de uma empresa (produção e vendas). O método utilizado para se definir onde se deve cortar de forma geral começa com uma seleção dos projetos orçados que ainda não tem seus objetivos e principalmente os benefícios plenamente esclarecidos e compreendidos por todos os executivos.

Por outro lado, a agilidade em redefinir estratégias e a rápida adaptação às mudanças são o grande diferencial competitivo das empresas líderes. Sem a informação adequada, no momento certo, as chances de sucesso de uma nova iniciativa diminuem e desta forma a solução encontrada pelas empresas para mitigar o risco da imobilidade corporativa muitas vezes está em ceder espaço às solicitações dos diferentes departamentos, gerando com isto novas demandas de informações e sistemas. Normalmente ações tomadas de forma rápida não obrigatoriamente significam ações de menor custo.

Este passa a ser o grande paradoxo das organizações: agilidade e performance corporativa versus manutenção e redução de custos. Em função deste paradoxo normalmente nas empresas a percepção das áreas de negócio em relação a TI é a de que a mesma é uma área que gasta muito e não consegue atender os requisitos de negócio no tempo adequado. Esta impressão é reforçada normalmente por problemas de comunicação e baixa interação entre as áreas.

Reforçar o compromisso entre o departamento de TI e as áreas de negócio é um dos fatores chave de sucesso no sentido de se conseguir a flexibilidade e agilidade na estruturação dos processos de negócio. Para se reforçar este compromisso é importante primeiramente se fazer entender, conseguindo explicar e de certa forma orientando as áreas sobre como TI pode contribuir para o negócio, desenvolvendo as soluções adequadas a um custo menor e com mais agilidade. A visão adequada de arquitetura de sistemas neste caso pode ser o primeiro passo para se atender a estes requisitos, na medida em que a mesma orienta as soluções de TI e pode reforçar o entendimento das áreas de negócio quanto ao retorno esperado pelas soluções propostas. Projetar esta arquitetura de forma adequada é fator preponderante para se conseguir a tão almejada agilidade, e é aí neste momento, que o conceito de SOA pode ser usado como o elo que une todas estas pontas.

O conceito de SOA é relativamente recente e tem despertado o interesse de muitas organizações no mundo inteiro. Atualmente há uma grande quantidade de material sobre SOA, mas sem uma visão integrada deste tipo de arquitetura . A principal dificuldade das empresas é ter um modelo que estabeleça métodos consistentes para implementar e medir os resultados SOA, cuja lacuna esta monografia pretende preencher.

1.3. Abrangência

O foco central do trabalho são as fases de análise, especificação e desenvolvimento da arquitetura SOA. As fases de implementação e gerenciamento pós-implementação não são focos desta monografia, assuntos que podem ser abordados em trabalhos futuros. Apesar do foco ser direcionado no desenvolvimento da arquitetura, não se pode deixar de lado as estratégias, metas e objetivos organizacionais, bem como o alinhamento destas metas e objetivos com a visão estabelecida da arquitetura corporativa orientada a serviços a ser desenvolvida.

Mais importante do que a existência deste alinhamento entre objetivos e estratégias organizacionais com a arquitetura corporativa de TI é a mensuração dos benefícios a serem obtidos com a implementação deste tipo de arquitetura. Para isto é proposto o modelo Balanced Scorecard de KAPLAN & NORTON (1997) , onde é possível estabelecer indicadores mensuráveis para cada objetivo ou benefício, possibilitando também o cálculo do Retorno do Investimento (ROI) com a implementação SOA, apesar de não entrar em detalhes de engenharia financeira, que faz parte da disciplina de administração financeira.

2. REVISÃO DA LITERATURA

2.1. Arquitetura Corporativa de TI

BOTTO(2004) apresenta a definição de Arquitetura Corporativa como "O conjunto estruturado de dados e modelos descritivos que definem o negócio, a informação e a tecnologia suportada para operar a empresa, num processo contínuo de alinhamento com o negócio."

GEAO (2006) nos fornece a seguinte definição: "Arquitetura Corporativa é o modo pelo qual uma visão corporativa é expressa na estrutura e dinâmica de uma corporação. Ela provê, em vários níveis de abstração de arquitetura, um conjunto coerente de modelos , princípios, guias e diretrizes usadas na tradução, alinhamento e evolução de sistemas existentes dentro do escopo e do contexto de uma empresa."

2.1.1. Principais *Frameworks*[1] de Arquitetura Corporativa

Os *frameworks* de arquitetura corporativa têm um papel fundamental para descrever a arquitetura corporativa. Ainda segundo BOTTO (2004), "Podemos dizer que um *framework* de arquitetura é um modelo para que uma organização realize um mapeamento de seu acervo de sistemas e bases de TI, sendo agrupado segundo uma certa classificação, conforme o grupo que organizou o framework."

O primeiro *framework* de arquitetura corporativa de TI foi idealizado por John Zachman que, em 1987, publicou seu framework que se tornaria base para outros que evoluíram a partir do seu modelo. Vários outros surgiram na década de 90, tais como:

- FEAF do Conselho de CIO norte-americano ;
- FEA-PMO do órgão federal americano OMB ;
- TEAF do órgão do Tesouro Americano ;

[1] Um *framework* de arquitetura divide a enorme quantidade de detalhe de uma empresa em pedaços gerenciáveis.

- DoDAF do departamento de defesa americano ;
- TOGAF do Open Group.

O principal deles, o Framework de Zachman, é descrito a seguir. Zachman construiu uma matriz corporativa onde as linhas são visões da organização, as colunas são aspectos que referenciam de alguma forma a empresa. As células desta matriz são as visões da organização em cada um destes aspectos. É um esquema de classificação genérica para construção de artefatos. O objetivo é habilitar o usuário a focar em aspectos selecionados da empresa sem perder o contexto. Divide a enorme quantidade de detalhes de uma empresa em partes gerenciáveis , o qual está melhor ilustrado na Figura 2.1 abaixo.

	O Quê	Como	Onde	Quem	Quando	Por quê
	Dados	**Função**	**Rede**	**Pessoal**	**Tempo**	**Motivação**
ESCOPO (Contextual)	Lista de Coisas Importantes para o Negócio	Lista de Processos que o Negócio Realiza	Lista de Locais onde o Negócio Opera	Lista de Organizações Importantes para o Negócio	Lista de Eventos Importantes para o Negócio	Lista de Metas e Estratégias
Planejador						
MODELO DE NEGÓCIOS (Conceitual)	Modelo Semântico	BPM – Modelo de Processos do Negócio	Sistema de Logísticas do Negócio	Workflows	Cronograma Mestre	Plano de Negócios
Dono						
MODELO DE SISTEMAS (Lógico)	Modelo Lógico de Dados	Arquitetura de Aplicações	Arquitetura de Sistemas Distribuídos	Arquitetura de Interface Homem – Máquina	Estrutura de Processamento	Modelo de Regras de Negócios
Designer						
MODELO DE TECNOLOGIA (Físico)	Modelo Físico de Dados	Design de Sistemas	Arquitetura de Tecnologia	Arquitetura de Apresentação	Estrutura de Controle	Design de Regras
Construtor						
REPRESENTAÇÕES DETALHADAS	Definição de Dados	Programa	Arquitetura de Redes	Arquitetura de Segurança	Definição de Ciclos	Especificação de Regras
EXEMPLO	Dados	Função	Rede	Organização	Agenda	Estratégia

Figura 2.1 – Adaptado do *Framework* Zachman de Arquitetura Corporativa (ZIFA, 2006)

2.1.2. Benefícios

A seguir BOTTO(2004) lista alguns benefícios identificados ao se implantar, em uma empresa, um framework de Arquitetura Corporativa:

- Benefícios estratégicos: direcionamento estratégico, adaptação à mudança, atendimento a requisitos legais e regulatórios e aumento da disponibilidade dos sistemas.
- Benefícios de processos: alinhamento com processo de negócio , comunicação entre a área de negócios e TI e alinhamento com parceiros.
- Benefícios de governança de TI: melhor planejamento, otimização dos recursos de TI.
- Benefícios no desenvolvimento e manutenção: diminuição nos prazos de entrega , minimização dos riscos associados a cada projeto de desenvolvimento e facilita a confecção de guias e na substituição de tecnologias e produtos obsoletos.

2.1.3. Tipos de Arquitetura Corporativa

A seguir são descritas os principais tipos de arquitetura que irão compor o conjunto total de mapas e modelos da arquitetura corporativa, segundo BOTTO (2004):

- **Arquitetura de Negócios**: corresponde às áreas e processos de negócios corporativos. Nesta arquitetura os principais serviços e produtos que a empresa oferece precisam estar modelados em termos de processo e regras de negócio.
- **Arquitetura de Informação**: modela o conteúdo dos processos de negócios pertinentes à empresa. Junto com a Arquitetura de Negócios, a Arquitetura de Informação dará subsídios para que a Arquitetura de Sistemas seja mapeada e

permita também atender processos de negócios corporativos, no escopo de arquiteturas futuras.

- **Arquitetura de Sistemas**: mapa corporativo das aplicações e de suas interfaces, onde seus dados estejam todos correlacionados, se relacionando estreitamente com as Arquiteturas de Informação e de Integração.
- **Arquitetura de Integração**: também chamada de *middleware*[2] corporativo, descreve todos os meios de relacionamento entre os diversos componentes implementados, e portanto, descritos na Arquitetura Tecnológica sendo um elo entre as arquiteturas superiores e esta última.
- **Arquitetura de Tecnologia**: é a camada mais próxima do operacional e da infra-estrutura fazendo parte o hardware, software, ferramentas, redes, segurança, produtos presentes na planta técnica de TI.
- **Arquitetura de Dados**: lida com a estrutura de banco de dados, qualidade e performance de acesso aos dados, estabelecendo padrões e políticas de uso e acesso à informação corporativa, aspectos operacionais de consolidação, replicação, proteção, migração, carga e assim por diante.
- **Arquitetura de Plataformas**: lida com hardwares, sistemas monitores, sistemas operacionais, armazenamento, backup, sendo que uma infra-estrutura de plataformas bem modelada permite medir impactos significativos em mudanças de sistemas ou processos de negócios.
- **Arquitetura de Redes**: rege a conectividade entre elementos da Arquitetura Tecnológica, visando troca eletrônica de dados , distribuição, suporte, operação entre os ambientes que precisam ser atendidos para os objetivos corporativos.

[2] Middleware é o software que integra outros softwares, ou sistemas.

- **Arquitetura de Segurança**: tem como finalidade a proteção da informação corporativa.

2.2. Arquitetura Orientada a Serviços (SOA)

2.2.1. Definição

O termo *Service-Oriented Architecture* foi inicialmente proposto por SCHULTE & NATIS (1996), então analistas do Gartner. Eles definiram SOA como "uma configuração de computação multicamadas que ajudam as organizações compartilhar lógica e dados através de múltiplas aplicações e modelos de uso".

Imagine estar preparado para atender às demandas das áreas de negócios com mais agilidade e flexibilidade. Coloque como cenário um ambiente baseado em uma arquitetura de software modular, onde todos os aplicativos são acessados por uma única interface web e os sistemas utilizam dados uns dos outros e "conversam" indiscriminadamente. Isto é o pano de fundo de uma arquitetura orientada a serviços (SOA). Neste novo modelo arquitetural, não há mais uma interface, um banco de dados e um sistema de integração para cada software. Na web, em uma única tela, os usuários acessam todos os programas de forma transparente. Além de mais flexível e mais amigável, a interface única dá muito mais agilidade e transparência ao usuário. Os aplicativos, por sua vez, utilizam qualquer informação, independentemente de onde ela tenha sido gerada. Graças ao mapeamento lógico dos dados, é como se todos estivessem em uma única base, ajudando a reduzir a inconsistência.

A *Organization for the Advancement of Structured Information Standards* (OASIS) 2006, define SOA da seguinte forma:

Arquitetura Orientada a Serviços (SOA) é um paradigma para organização e utilização de competências distribuídas as quais estão sob o controle de diferentes domínios proprietários. Ela fornece um meio uniforme de oferecer, descobrir, interagir e usar capacidades para produzir efeitos desejáveis consistentes com pré-condições e expectativas mensuráveis.

Para exemplificar, a seguir é apresentado, segundo o Modelo de Referência para SOA de OASIS (2006) , um exemplo trabalhado de Arquitetura Orientada a Serviços:

*Uma empresa de eletricidade tem a capacidade de gerar e distribuir eletricidade (**capacidade subjacente**). A fiação da rede de distribuição da companhia elétrica (**o serviço**) oferece o meio para fornecer eletricidade para suportar o uso por um consumidor residencial típico (**funcionalidade do serviço**), e um consumidor acessa a eletricidade gerada (**a saída da invocação de serviço**) via uma tomada de parede (**interface de serviço**). De forma a utilizar a eletricidade, um consumidor precisa entender que tipo de plug usar, qual a voltagem fornecida e quais os possíveis limites de carga ; a empresa presume que o consumidor irá conectar somente aparelhos adequados à voltagem ofertada e à carga suportada ; e o consumidor por sua vez, assume que os aparelhos adequados podem ser conectados sem danos ou riscos (**suposições técnicas do serviço**).*

*Um usuário residencial ou comercial precisa abrir uma conta na empresa para usar o fornecimento (restrição de serviço) e a empresa irá medir o consumo e espera que o consumidor pague pela energia conforme taxa prevista (**política de serviço**). Quando o consumidor e a empresa concordam nas restrições e políticas (**contrato de serviço**), o consumidor pode ter o fornecimento de eletricidade usando o serviço desde que a rede de distribuição de eletricidade e a conexão residencial permaneçam intactas (por exemplo, uma tempestade que derrube a rede e interrompa o fornecimento) e o consumido pode pagar (por exemplo, transferência eletrônica de fundos) a empresa (**acessibilidade**). Em certas situações (por exemplo, demanda excessiva), uma empresa pode limitar o fornecimento ou*

instituir cortes rotativos (políticas de serviço). Um consumidor pode guardar uma aceitação formal se isto ocorre freqüentemente (política implícita do consumidor). Se a empresa requer que cada aparelho seja firmemente ligado aos seus equipamentos, a capacidade subjacente pode ainda estar lá, mas isto pode ser um serviço diferente e ter uma ***interface de serviço diferente****.*

2.2.2. Benefícios e desafios de uma arquitetura SOA

2.2.2.1. Benefícios SOA

Os benefícios para interessados em implementar um SOA para gerenciamento da informação são significantes. Segundo SELVAGE et al. (2005), os seguintes benefícios são alcançados :

- ***Permite a reutilização dos ativos de TI***: Este é o primeiro e o mais importante benefício de implementar SOA. Pode-se construir um serviço de negócio como uma agregação de componentes existentes. Para utilizar este novo serviço é exigido somente saber seu nome e interface. A implementação de um serviço específico, bem como sua arquitetura, bem como as complexidades do fluxo de dados que compõe este serviço, são transparentes para quem o chama. Isto permite às organizações aproveitar investimentos atuais, construir serviços de um conglomerado de componentes construídos em diferentes plataformas, que rodam em diferentes sistemas operacionais e desenvolvidos em diferentes linguagens de programação. Sistemas legados podem ser encapsulados e acessados utilizados as *interfaces web services*.
- ***Diminui o tempo de desenvolvimento e reduz os custos de desenvolvimento e manutenção***: como as demandas do negócio crescem e novos requisitos são introduzidos a todo momento, o custo de desenvolver e criar novos serviços através do SOA framework e

biblioteca de serviços, o custo é bastante reduzido. A curva de aprendizagem de uma equipe de desenvolvimento é também reduzida, pois já podem estar familiarizados com os componentes existentes.

- ***Diminuição do risco:*** reusar componentes existentes reduzem o risco de introduzir novas funcionalidades dentro de processos de melhoria ou criar novos serviços do negócio. O risco de manutenção e gerenciamento da infra-estrutura dos serviços de suporte também será reduzida.
- ***Proteção do investimento:*** protege o investimento do cliente no gerenciamento da informação que é altamente volátil num mercado onde as fusões e aquisições são freqüentes.

2.2.2.2. Desafios SOA

Atualmente a Tecnologia da Informação tem sofrido várias modificações, impulsionada pela necessidade de melhoria no relacionamento que existe entre as áreas que suportam tecnologicamente os negócios e os negócios propriamente ditos. Seus gestores estão se dando conta da necessidade e da conveniência do enfoque nos serviços, como um fim, sendo vitais para os negócios em um futuro que já se tornava atualidade. A SOA trouxe à tona a necessidade de fortalecer o enfoque no cliente e tornar a gestão de serviços como uma atividade produtiva, que gere valor à empresa.

Como tal, essa atividade é fortemente dependente das pessoas e a caminhada rumo a SOA é árdua, exigindo um *forte investimento na evolução organizacional*, na conscientização da necessidade de evoluir e mudar, no estabelecimento de um gerenciamento de pessoas eficaz, orientada a conhecer a potencialidade, objetivos e desejos das pessoas em detrimento dos objetivos da organização, orientando à gerência de serviços em acordo com o desempenho individual. Tornar os desafios da organização aderentes as pessoas é um dos maiores desafios da Gestão da mudança

organizacional. Dessa forma SOA possibilita a criação de novos aplicativos com maior coerência, rapidez e diminuição nos custos, tudo isso com excelente aproveitamento do legado.

2.2.3. Principais termos e definições de SOA

A ***visibilidade, interação e efeitos*** são os conceitos chaves para descrever o paradigma SOA, segundo OASIS (2006). A ***visibilidade*** refere-se à capacidade para aqueles com necessidades e aqueles com competências estarem aptos a se verem mutuamente. Isto é possível pelo oferecimento de descrições acerca destes aspectos como as funções e requisitos técnicos , restrições e políticas relacionadas , e mecanismos para acesso e resposta. Enquanto a visibilidade introduz a possibilidade de compatibilizar as necessidades com as competências (e vice-versa), a ***interação*** é a atividade que usa a competência Mediada por troca de mensagens , uma interação prossegue através de uma série de ações de troca de informações e invocações. O propósito de usar as competências é realizar um ou mais ***efeitos*** no mundo real. Como principal, uma interação é "um ato" em oposição à "Um objeto" e o resultado de uma interação é um efeito (ou um conjunto/série de efeitos). Este efeito pode ser o retorno de uma informação ou a mudança no estado de entidades (conhecidas ou desconhecidas) que estão envolvidas na interação.

Esta descrição do SOA tem ainda que mencionar o que é usualmente considerado o conceito central: **o serviço**. O termo "serviço" é definido no dicionário como "O desempenho de trabalho (uma função) por alguém para outro." Contudo, serviço, como o termo é geralmente entendido, também combina as idéias relacionadas com:

- A competência de executar o trabalho para outro ;
- A especificação do trabalho oferecido para outro ;
- A oferta para executar trabalho para outro.

Serviço é uma função independente, sem estado , que aceita uma ou mais requisições e retorna uma ou mais respostas através de uma interface padronizada e bem definida. Serviços podem também realizar partes

discretas de um processo tal como editar ou processar uma transação. Serviços não devem depender do estado de outras funções ou processos. A tecnologia utilizada para prover o serviço, tal como uma linguagem de programação, não pode fazer parte da definição do serviço.

ERL (2005) estabelece uma relação entre serviço e operações, como ilustrado na Figura 2.2 abaixo, onde pode-se observar que um serviço representa um conjunto logicamente agrupado de operações capaz de executar relacionados unidades de trabalho.

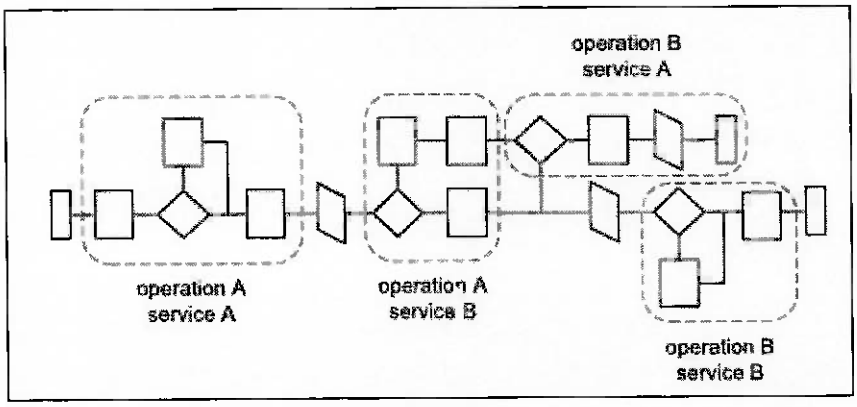


Figura 2.2 - Operações pertencentes a diferentes serviços representam várias partes de um processo de negócios lógico (ERL, 2006)

Termo	Definição / Comentário
Serviço	É uma função independente, sem estado (*stateless*) que aceita uma ou mais requisições e retorna uma ou mais respostas através de uma interface padronizada e bem definida. Serviços podem também realizar partes discretas de um processo tal como editar ou processar uma transação. Serviços não devem depender do estado de outras funções ou processos. A tecnologia utilizada para prover o serviço, tal como uma linguagem de programação, não pode fazer parte da definição do serviço.
Orquestração	Processo de seqüenciar serviços e prover uma lógica adicional para processar dados. Não inclui uma representação de dados.

Termo	Definição / Comentário
Stateless	Não depende de nenhuma condição pré-existente. Os serviços não devem depender de condições de outros serviços. Eles recebem todas as informações necessárias para prover uma resposta consistente. O objetivo de buscar a característica de *stateless* dos serviços é possibilitar que o consumidor do serviço possa seqüenciá-lo, ou seja, orquestrá-los em vários fluxos (algumas vezes chamados de *pipelines*) para executar a lógica de uma aplicação.
Provedor	O recurso que executa o serviço em resposta a uma requisição de um consumidor.
Consumidor	É quem consome ou pede o resultado de um serviço fornecido por um provedor.
Descoberta	SOA se baseia na capacidade de identificar serviços e suas características. Conseqüentemente, esta arquitetura depende de um diretório que descreva quais os serviços disponíveis dentro de um domínio.
Binding	A relação entre os serviços do provedor e do consumidor deve ser idealmente dinâmica; ela é estabelecida em tempo de execução através de um mecanismo de *binding*.

Tabela 2.1 - Principais definições de SOA

2.2.4. Princípios SOA

ERL (2005) define os seguintes princípios de uma arquitetura orientada a serviços:

- **Reusabilidade do serviço** – A lógica é dividida dentro dos serviços com a intenção de promover o seu reuso.

- **Fraco acoplamento** – Serviços mantêm um relacionamento que minimiza dependências e somente requer que eles mantenham o conhecimento da existência um do outro..
- **Contrato de serviço** – Os serviços aderem a um acordo da comunicação, como definidos coletivamente por um ou mais documentos de descrição do serviço.
- **Abstração** – Além do que está descrito no contrato de serviços, serviços escondem a lógica para quem está olhando de fora.
- **Composabilidade** – Coleções de serviços podem ser coordenados e montados para formar o menu de serviços.
- **Autonomia** – Serviços têm o controle sobre a lógica que eles encapsulam.
- **Sem estado** – Serviços minimizam a retenção da informação específica para uma atividade.
- **Descobertabilidade** – Os serviços são projetados para serem descritivos de modo que podem ser achados e podem ser avaliados via mecanismos disponíveis de descoberta.

Todos os princípios são bastante importantes, mas o da reusabilidade passa a ser o principal , conforme ilustrado na Figura 2.3 abaixo. Quando um serviço encapsula a lógica que é utilizada por mais do que uma requisição de serviço, este pode ser considerado reusável. O conceito de reuso é baseado por um número de princípios complementares, tais como:

- **Autonomia do serviço** estabelece um ambiente de execução que facilita o reuso porque o serviço tem independência e auto-governança.
- **Sem-estado do serviço** suporta o reuso porque ele maximiza a disponibilidade de um serviço e em geral promove uma especificação genérica do serviço que difere do processamento de atividades específicas fora das fronteiras lógicas do serviço.

- **Abstração do serviço** permite o reuso porque estabelece o conceito de caixa preta, onde os detalhes do processamento são completamente escondidos do requisitante. Isto permite um serviço expressar de uma forma simples uma interface pública genérica.
- **Descobertabilidade** do serviço promove o reuso , e isto permite requisitantes procurar e descobrir serviços reusáveis.
- **Baixo acoplamento** dos serviços estabelece uma independência intrínsica que liberta um serviço das amarras imediatas de outros. Isto tem uma grande influência em deixar um serviço reusável.

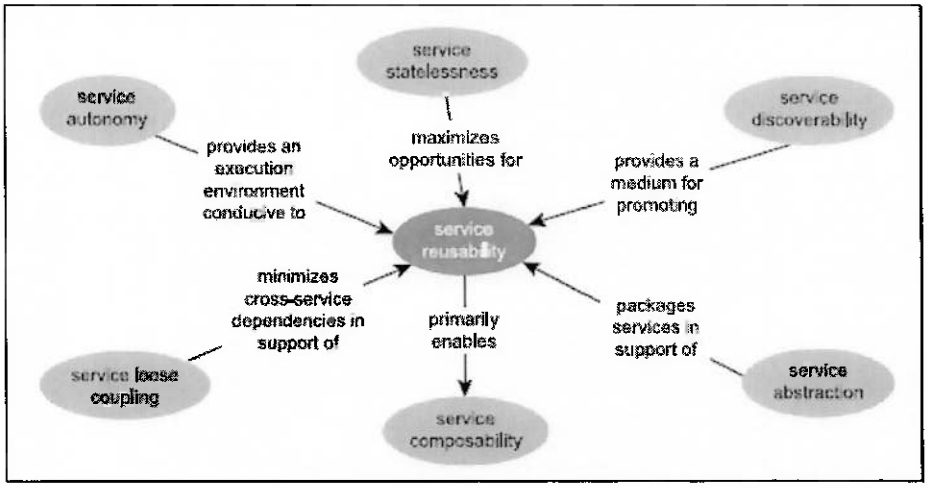


Figura 2.3 - Princípio da reusabilidade dos serviços e seu relacionamento com outros princípios SOA (ERL, 2006)

2.2.5. As Camadas de uma Arquitetura SOA

ERL (2005) apresenta um alto nível de abstração da camada de se serviços posicionada entre as camadas dos processos de negócio e de aplicação, que demonstra a conectividade aberta entre as camadas de serviço. No nível físico, serviços são desenvolvidos em ambientes proprietários, onde eles são individualmente responsáveis pelo encapsulamento de uma aplicação lógica específica.

A Figura 2.4 mostra como serviços individuais dentro da camada lógica de serviços , representam aplicações lógicas originadas de diferentes plataformas.

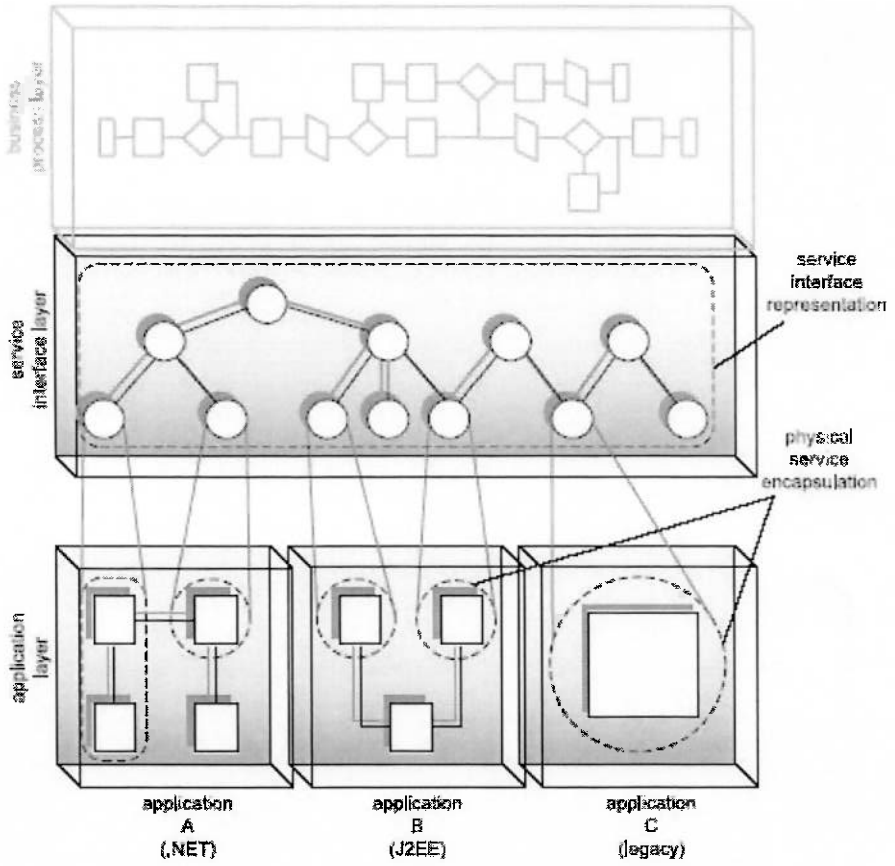


Figura 2.4 – Camada de serviços posicionada entre as camadas de processos de negócio e de aplicação (ERL, 2006)

2.2.6. SOA e Processos de Negócio

NOEL (2005) define que *Business Process Management* (BPM) é uma solução que modela, monitora, simula e redesenha processos de negócio, permitindo maior flexibilidade e agilidade aos negócios. Para que isto aconteça, os processos devem tornar-se independentes de uma específica fonte de informação e aplicação, ou seja, a tecnologia de integração deve "acoplar fracamente" as aplicações e recursos que formam os processos, caso contrário a

lógica do processo ficará "amarrada" à uma específica plataforma tecnológica, o que pode tornar difícil e elevar os custos de uma eventual mudança. SOA vem de encontro à esta necessidade, pois fornece a habilidade técnica para criar a independência dos processos. Assim, processos modelados com ferramentas BPM podem ser mais facilmente implementados em uma arquitetura SOA.

Algumas típicas funcionalidades de uma solução de BPM incluem:

- Métricas de progresso e performance de processos-chave;
- Atribuição de papéis organizacionais aos processos, colaboração entre os papéis e habilidades pessoais;
- Automação de funções de negócio por aplicações específicas ou serviços;
- Regras de negócio e relacionamentos externos à organização
- Seqüenciamento de atividades e dependências de sub-processos.

Como acontece em muitas organizações, a comunicação entre aplicações e entre aplicações e usuários são altamente dependente das interfaces e estrutura de dados das aplicações que vão sendo integradas. O resultado é a integração *spaghetti* conforme mostra a Figura 2.5. Este forte acoplamento significa que mudanças em uma das aplicações requer mudanças também na solução de integração. Significa também que se algum processo é alterado, novos *links* entre diferentes aplicações devem ser construídos.

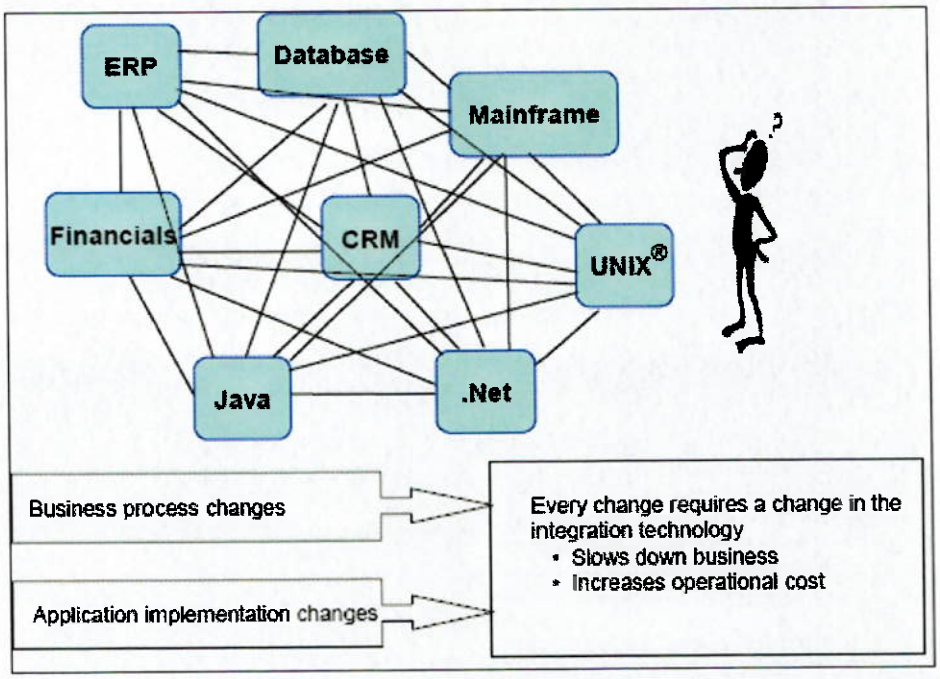


Figura 2.5 – Integração com acoplamento forte não permite rapidamente se adaptar às mudanças do negócio (NOEL, 2005)

Um dos objetivos da arquitetura SOA é justamente eliminar a integração *spaghetti* que existem na maioria das empresas atualmente, através do reuso dos serviços. SOA fornece a evolução que a integração requer com a habilidade de criar a independência necessária entre processos e a implementação do serviços, independentemente da plataforma tecnológica, resultando assim no que é chamado de integração com acoplamento fraco, conforme mostra a Figura 2.6 abaixo.

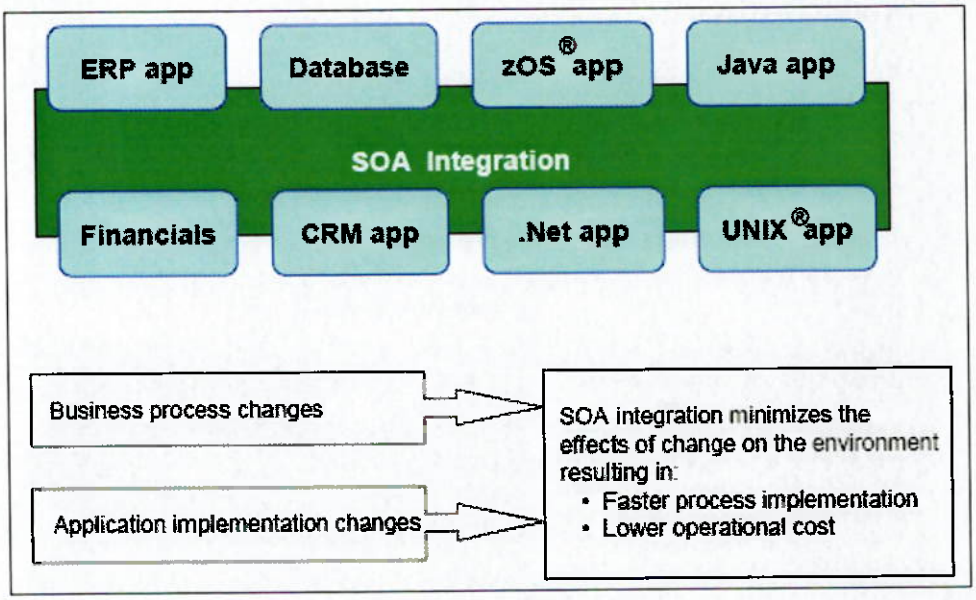


Figura 2.6 – Integração com acoplamento fraco permite rapidamente se adaptar às mudanças do negócio (NOEL, 2005)

Criando esta independência processo/serviço facilita num melhor alinhamento entre modelagem de processos de negócio e sua implementação. Processos novos e alterados que são modelados em BPM podem ser implementados na infra-estrutura organizacional mais rapidamente pois a solução SOA desacopla a especificação do processo de sua específica implementação, conforme mostra Figura 2.7 abaixo.

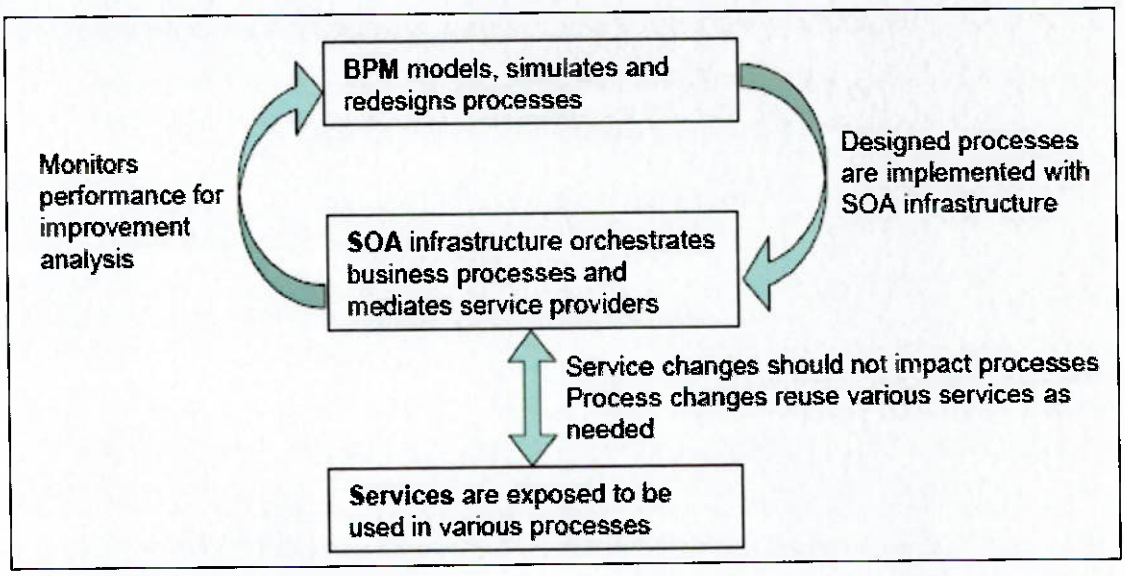


Figura 2.7 – Relacionamento entre BPM e SOA (NOEL, 2005)

2.2.7. SOA e a Tecnologia *Web Services*

SOA expressa um conceito onde aplicativos ou rotinas são disponibilizadas como serviços em uma rede de computadores (Internet ou Intranets) de forma independente e se comunicando através de padrões abertos. A maior parte das implementações de SOA se utilizam de Web services. Entretanto, uma implementação de SOA pode se utilizar de qualquer tecnologia padronizada baseada em web.

A base do padrão Web services relevantes ao SOA incluem:

- **XML (*eXtensible Markup Language*)** – é uma linguagem de marcação para descrever dados em cargas de mensagens em um formato de documento.
- **SOAP (*Simple Object Access Protocol*)** - é um protocolo baseado em XML utilizado para trocas de informações num ambiente distribuído. A utilização de XML é uma forma eficiente para trocas de informações mas não é suficiente para trocas de dados via Web. Ainda precisamos de um protocolo em comum para envio de documentos XML pela Web de tal forma que o receptor entenda o que esta recebendo e o que deve ser respondido. É neste ponto que entra o SOAP.
- **WSDL (*Web Services Description Language*)** - é um documento XML que descreve um conjunto de mensagens SOAP e a forma como essas mensagens são trocadas. Como o WSDL é XML, ele é legível e editável, mas na maioria dos casos, ele é gerado e consumido pelo software. Para enxergar o valor do WSDL, imagine que você quer invocar um método SOAP que é fornecido por um de seus parceiros de negócios.
- **UDDI (*Universal Description, Discovery, and Integration)*** - UDDI é uma das formas de localizar um WebService num registro, como se fosse um catálogo de páginas amarelas, de tal forma que um programa em busca de um determinado serviço possa facilmente localizar e entender o que ele faz.

Segundo ARSANJANI (2004) a Arquitetura Orientada a Serviços pode ser bem representada a partir do seguinte modelo envolvendo três partes principais : o fornecedor do serviço, o usuário e o registro do serviço , também chamado de "find-bind-execute paradigm " o que significa paradigma de "procurar-consolidar-executar" conforme melhor demonstrado na Figura 2.8. Tal conceito é análogo ao "Ciclo de Deming" aplicado aos serviços, que define o ciclo que envolve o planejamento, a execução, o monitoramento e a tomada de ação pró-ativa para a melhoria da qualidade.

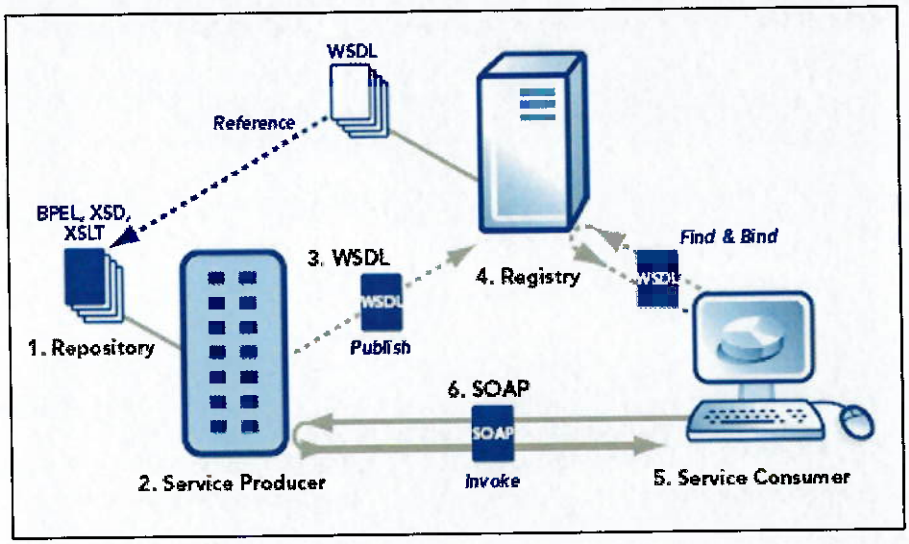


Figura 2.8 - Paradigma "procura-consolida-executa" (adaptado de ARSANJANI, 2004)

Tecnicamente falando, o processo preconiza que os provedores de serviços registrem informações em um registro central, com suas características, indicadores, e aspectos relevantes às tomadas de decisões. O registro é utilizado pelo cliente para determinar as características dos serviços necessários, e se o mesmo estiver disponível no registro central, como por exemplo, por um catálogo de serviços, o cliente poderá utilizá-lo, sendo este oficializado através de um contrato de serviço que enderece este serviço.A comunicação entre fornecedor e consumidor do serviço segundo o paradigma "procura-consolida-executa" é realizado através de um Contrato de Serviço o qual necessita ter os seguintes componentes:

1. Cabeçalho

- Nome do serviço: deve indicar em termos gerais o que faz, mas não é a única definição.
- Versão – a versão do contrato de serviço
- Proprietário – a pessoa/equipe que gerencia o serviço
- RACI
    1. Responsável – pessoa/equipe responsável para as entregas deste contrato/serviço.
    2. *Accountable*(Decisor) – decisor final para alterar o contrato de serviço.
    3. Consultor - quem deve ser consultado antes de alguma alteração ser realizada no contrato de serviço.
    4. Informados - quem deve ser informado da decisão a ser tomada. Esta é uma comunicação de uma via. Estas pessoas são impactadas pela decisão ou execução desta decisão, mas não têm controle sobre a ação.
- Tipo – este tipo de serviço ajuda a distinguir a camada que o serviço reside:
    1. Dado
    2. Processo
    3. Funcionalidade
    4. Apresentação

2. Funcional
    - Requisito funcional (Do documento de requisito funcional) – indica a funcionalidade em itens específicos e o que exatamente este serviço executa.
    - Operações do serviço – métodos, ações devem ser definidas em termos do que esta funcionalidade fornece.
    - Invocação - indica o meio de invocação do serviço. Isto inclui a URL, interface, havendo talvez múltiplos diretórios de invocação para o mesmo serviço. Podemos ter a mesma funcionalidade para clientes externos e internos cada um com um meio diferente de invocação e interface.

3. Não-Funcional
    - Variáveis de segurança – define que executa este serviço em termos de papéis de parceiros individuais e qual mecanismo de chamada ele podem ter.
    - Qualidade do serviço – determina a taxa admissível de falha do serviço.
    - Transacional – este serviço é capaz de atuar como parte de uma transação maior e se sim, como nós controlamos isso ?
    - Acordo de Nível de Serviço – determina a valor da latência do serviço é permitido para executar as ações.
    - Semântica - define o significado de termos usados na descrição e interfaces do serviço .
    - Processo – descreve o processo do serviço contrato.

2.2.8. Estratégias de desenvolvimento e implementação SOA

ERL (2005) define as seguintes estratégias para desenvolvimento e implementação SOA:

- **A estratégia *top-down***

Esta estratégia de *cima para baixo* requer não somente que os processos de negócio se tornem orientados-a-serviço, mas também promove a criação (ou realinhamento) de todo o modelo de negócios da organização. Os seguintes passos estão previstos nesta abordagem:

1. Definir processos de negócio
2. Composição SOA
3. Definir o modelo de serviço
4. Análise orientada-a-serviço
5. Especificação orientada-a-serviço
6. Desenvolvimento dos serviços
7. Testes
8. Implementação

- **A estratégia *bottom-up***

Esta abordagem *de baixo para cima* encoraja primeiramente a criação das aplicações de serviços servindo como base para as demais fases e integração passa a ser o motivador principal desta estratégia. Os seguintes passos estão previstos nesta abordagem:

1. Modelar as aplicações de serviços
2. Especificar as aplicações
3. Desenvolvimento das aplicações
4. Testes
5. Implementação

A escolha de uma ou outra estratégia vai depender de cada organização considerando as variáveis tempo e recursos financeiros que dispõe. A abordagem *de cima para baixo* claramente exige mais tempo e tem um custo bem mais alto que a abordagem de *baixo para cima*, pois esta última se utiliza das configurações e processos que se encontram na organização, não sendo necessário rever todos os processos e pode-se também escolher determinadas aplicações para torná-las orientadas-a-serviço.

- **A estratégia mista**

Pode-se também aplicar uma ***estratégia mista*** , ou seja, uma combinação entre as abordagens *top-down* e *bottom-up*, onde as fases de modelagem dos processos e do negócio se dão concomitantemente com as fases de especificação e implementação, necessitando um re-alinhamento com os processos em cada uma das fases.

2.3. Governança de TI

2.3.1. Definição

A governança de tecnologia de informação, ou governança de TI, é uma disciplina dentro da Governança corporativa focada aos sistemas de informação e seu desempenho e gerência de risco. O interesse ascendente em governança de TI é devido a iniciativas de conformidade (Sarbanes-Oxley (EUA) e Basiléia II (Europa)), assim como o reconhecimento que TI projeta facilmente pode ficar sem controlado e profundamente afetar o desempenho de um organização.

Um tema característico de discussões de governança de TI é que projetos de TI não pode mais ser uma caixa preta. A manipulação tradicional de projetos de TI por executivos é que pela experiência técnica e pela complexidade de projeto, decisões chave são submetidas a profissionais de TI. Com a governança de TI no entanto, todos interessados, incluindo o comitê executivo, clientes internos e áreas relacionadas tal como finanças, são exigidas participar no processo de tomada de decisão. Isto força todo o mundo aceitar responsabilidade por sistemas críticos e um único interessado, geralmente TI, será culpada por decisões pobres.

As metas principais para a governança de TI são:

1. garante que os investimentos em TI em geram valor para o negócio ;

2. mitigam os riscos que são associados com TI. Isto pode ser feito pela implementação de uma estrutura organizacional com papéis bem-definidos para a responsabilidade de informação, processos de negócio, aplicações, infra-estrutura e assim por diante.

Depois que o colapso largamente informado da Enron em 2000, e os supostos problemas dentro da Arthur Andersen e WorldCom, os deveres e responsabilidades dos comitês executivos de diretores para público e corporações em particular seguradas foram questionados. Como uma resposta a isto, e tentar prevenir problemas semelhantes aconteçam outra vez, a norma Sarbanes-Oxley dos EUA foi escrita para realçar a importância

de controle do negócio e auditoria. A Sarbanes-Oxley e Basiléia-II na Europa foram catalisadores para o desenvolvimento da disciplina de governança de TI desde o início de 2000.

Nicholas Carr emergiu como um crítico proeminente da idéia que a tecnologia da informação confere vantagem estratégica. Esta linha de crítica talvez encerre esse significativo atenção que a governança de TI não é uma busca vantajosa pela liderança corporativa. No entanto, Carr também indica interesse pela eficiente gerenciamento de risco de TI.

2.3.2. Modelos de Governança

Há um bom número mecanismos de suporte para desenvolver ou implementação de governança de TI. Algum deles são:

- ***IT Infrastructure Library* (ITIL)** é um detalhado framework de como alcançar sucesso em uma governança de TI , desenvolvida e mantida pelo desenvolvido e mantido pelo Escritório de Comércio de Governo do Reino Unido, em sociedade com o IT Forum de Gerência de Serviço.
- ***Control Objectives for IT* (COBIT)** é outra aproximação de padronização das boas práticas de segurança e controle. Isto é feito por fornecer ferramentas para avaliar e medir o desempenho de 34 processos de TI de uma organização. O Instituto de Governança de TI é responsável pelo CobIT.
- **ISO/IEC 27001 (ISO 27001)** é um conjunto de melhores práticas para organizações seguir e implementar e manter um programa de segurança. Começou com o Padrão britânico 7799 ([BS7799]), que foi publicado no Reino Unido e tornou-se um padrão conhecido na indústria que foi usado para fornecer direção a organizações na prática de segurança de informação.

Outros incluem:

- BS7799 – foco na segurança de TI .

- O CMM ou Modelo de Maturidade - tem foco em normas para a engenharia de software.

Frameworks não específicos de TI incluem:

- O ***Balanced Scorecard*** **(BSC)** – método para auxiliar a performance de uma organização em diferentes áreas, que utilizaremos neste método
- ***Six Sigma*** – foco na garantia da qualidade.

2.3.3. Balanced Scorecard

Balanced Scorecard é uma metodologia disponível e aceita no mercado. Foi desenvolvida pelos professores da Harvard Business School, Robert Kaplan e David Norton, em 1992. Os métodos usados na gestão do negócio, dos serviços e da infra-estrutura, baseiam-se normalmente em metodologias consagradas que podem utilizar a TI ou tecnologia da informação como solução de apoio, relacionando-a à gerência de serviços e garantia de resultados do negócio. Os passos dessas metodologias incluem: definição da estratégia empresarial, gerência do negócio, gerência de serviços e gestão da qualidade; passos estes implementados através de indicadores de desempenho.

Os requisitos para definição desses indicadores tratam dos processos de um modelo da administração de serviços e busca da maximização dos resultados baseados em quatro perspectivas que refletem a visão e estratégia empresarial, conforme ilustrado na Figura 2.9:

- financeira;
- clientes;
- aprendizado e crescimento;
- processos internos.

É um projeto lógico de um sistema de gestão genérico para organizações, onde o administrador de empresas deve definir e implementar (por exemplo, através de um Sistema de informação de gestão), variáveis de controle, metas e interpretações para que a organização apresente desempenho positivo e crescimento ao longo do tempo.

BSC (Balanced Scorecard) é uma sigla que pode ser traduzida para Indicadores Balanceados de Desempenho. O termo "Indicadores Balanceados" se dá ao fato da escolha dos indicadores de uma organização não se restringirem unicamente no foco econômico-financeiro, as organizações também se utilizam de indicadores focados em ativos intangíveis como: desempenho de mercado junto a clientes, desempenhos dos processos internos e pessoas, inovação e tecnologia. Isto porque, a somatória destes fatores, alavancarão o desempenho desejado pelas organizações, conseqüentemente criando valor futuro.

Segundo KAPLAN & NORTON (1997) , o *Balanced Scorecard* reflete o equilíbrio entre objetivos de curto e longo prazo, entre medidas financeiras e não-financeiras, entre indicadores de tendências e ocorrências e, ainda, entre as perspectivas interna e externa de desempenho. Este conjunto abrangente de medidas serve de base para o sistema de medição e gestão estratégica por meio do qual o desempenho organizacional é mensurado de maneira equilibrada sob as quatro perspectivas. Dessa forma contribui para que as empresas acompanhem o desempenho financeiro, monitorando, ao mesmo tempo, o progresso na construção de capacidades e na aquisição dos ativos intangíveis necessários para o crescimento futuro.

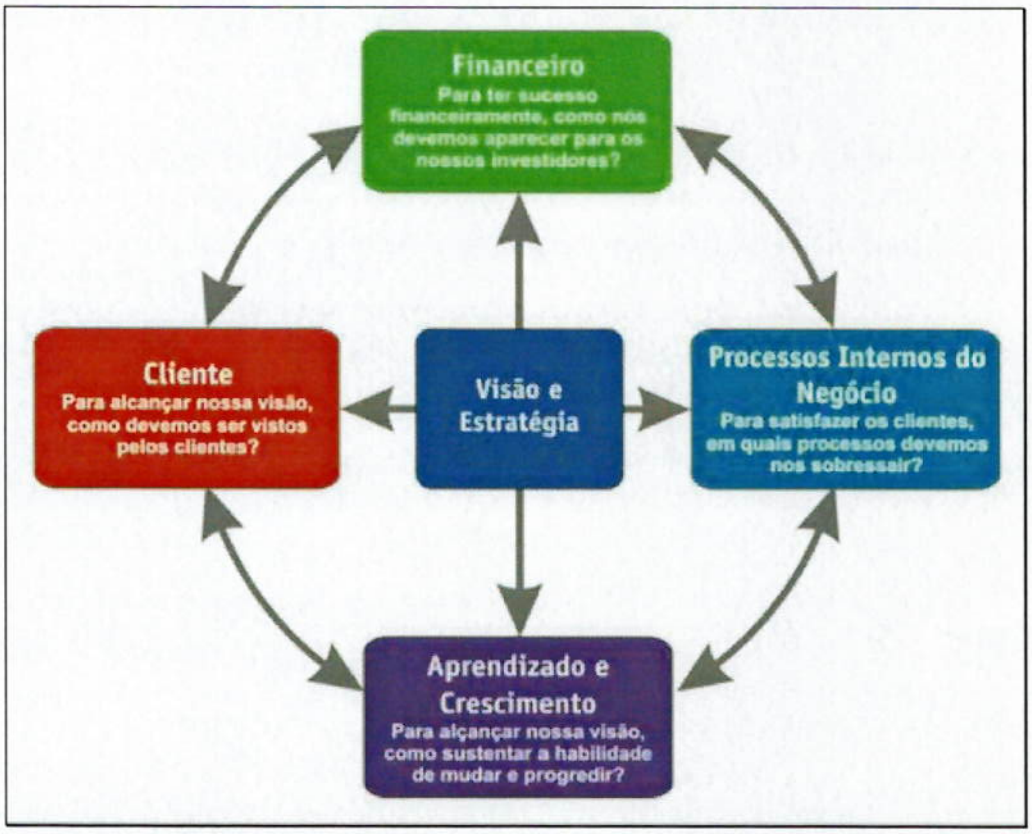


Figura 2.9 – Perspectivas do Balanced Scorecard (adaptado de KAPLAN & NORTON, 1997)

Portanto, a partir de uma visão balanceada e integrada de uma organização, o BSC permite descrever a estratégia de forma muito clara, através de quatro perspectivas: financeira; clientes; processos internos; aprendizado e crescimento. Sendo que todos se interligam entre si, formando uma relação de causa e efeito.

De acordo com a lógica estabelecida pelo método de KAPLAN & NORTON (1997), um balanced scorecard deve possuir os seguintes componentes, como ilustra a Figura 2.10 abaixo:

- Mapa estratégico
- Objetivo estratégico
- Indicador
- Meta
- Plano de Ação

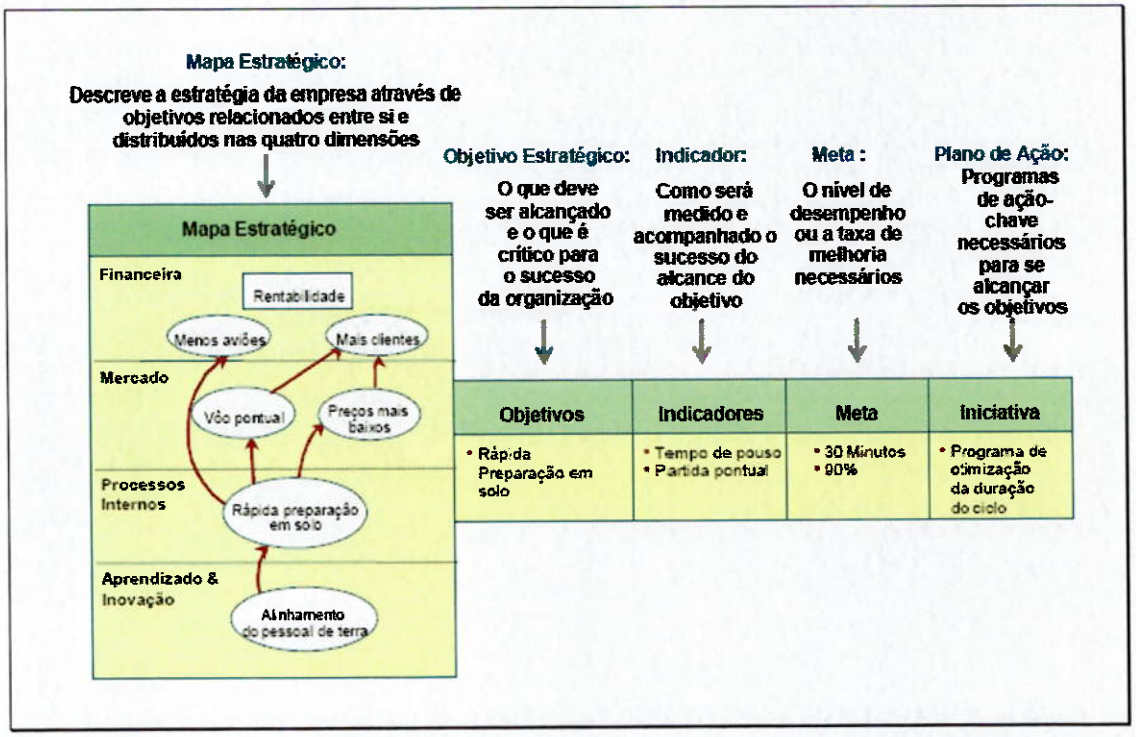


Figura 2.10 – Componentes e terminologia do balanced scorecard (adaptado de KAPLAN & NORTON, 1997)

Desde que foi criado, o BSC vem sendo utilizado por centenas de organizações do setor privado, público e em ONG's no mundo inteiro e foi escolhido pela renomada revista *Harvard Business Review* como uma das práticas de gestão mais importantes e revolucionárias dos últimos 75 anos.

2.3.4. ROI (*Return on Investment*)

ASSAF (2003) define que **Retorno sobre o Investimento (ROI)** é a relação de dinheiro ganho ou perdido num investimento de acordo com a quantia de dinheiro investido. A quantia de dinheiro ganho ou perdido pode ser referido ao lucro, lucro/perda, ganho/perda, ou ganho/perda líquida. O dinheiro investido pode ser referido ao ativo, capital, principal, ou a base de custo do investimento.

ROI normalmente é dado como um valor percentual. Por exemplo, um investimento de $1.000 investimento que ganha $50 em lucro obviamente gera mais dinheiro que um investimento de $100 que ganha $20 em lucro.

- $50/$1,000 = 5% de ROI
- $20/$100 = 20% de ROI

Em termos matemáticos , o ROI é definido como segue:

$$ROI = \frac{V_f - V_i}{V_i} = \frac{V_f}{V_i} - 1$$

onde: V_i é o investimento inicial e V_f é o valor final.

O ROI tem as seguintes características:

- *ROI* > zero quando o investimento é lucrativo
- *ROI* < zero quando o investimento tem perda

2.4. Exemplificação dos conceitos apresentados: a implementação de uma SOA no ING Card

A ING Card (INGC) 2006 apresentou um estudo de caso SOA de sua divisão do ING Group. Ele descreve as fases de análise, desenvolvimento e implementação SOA e ilustra como uma SOA pode entregar agilidade real ao negócio, e contém algumas lições interessantes para implementação SOA.

2.4.1. Introdução

ING Card recentemente entregou uma solução de automação que capacitou a instituição se ligar a novos *web sites*, implementar novas características de produto, e manter regras de pontuação de crédito, com facilidade e eficientemente. Isto foi alcançado por aplicar três princípios de construção, sendo um deles o de orientação-a-serviço. Este caso clínico descreve a aplicação e explica como uma SOA ajudou a encontrar as necessidades de negócio e desafios da administração de cartão de crédito da divisão de uma instituição financeira importante.

2.4.2. Desafios

Um negócio de cartão de crédito combina dois serviços financeiros: pagamento e crédito ao consumidor. O pagamento por plástico é aceito mundialmente e está ligado a uma rede grande de pagamentos, principalmente mantido pelas marcas MasterCard e VISA. O emissor de cartão normalmente também fornece uma facilidade de crediário que permite reembolso mais longo do saldo.

A divisão ING Card foi fundada como parte do Grupo ING em 2002, com o objetivo de centralizar o negócio de cartão de crédito do ING e se estendeu ao mercado europeu. O ING Card tornou-se um competidor

significativo no papel de emissor na complexa cadeia de valor de cartão de crédito (veja Figura 2.11).

O ING Card usa duas marcas de cartão de crédito, MasterCard e VISA, que possui redes mundiais de pagamento e fornece a 1 milhão de comerciantes (lojas, os hotéis, restaurantes, aluguéis de carro, etc) com equipamento de leitura de cartão. Os contratos de leitura normalmente são manipulados por *acquirers*[3] e isso facilita o acordo entre os comerciantes e as marcas de cartão de crédito. Outro elo na corrente é formado pelos processadores: organizações que executam as transações reais, fornecem a autorização de cartão no tempo de venda, e executa o acordo entre *acquirers*, marcas de cartão de crédito e emissores. Algumas instituições combinam os papéis de *acquirer* e processador.

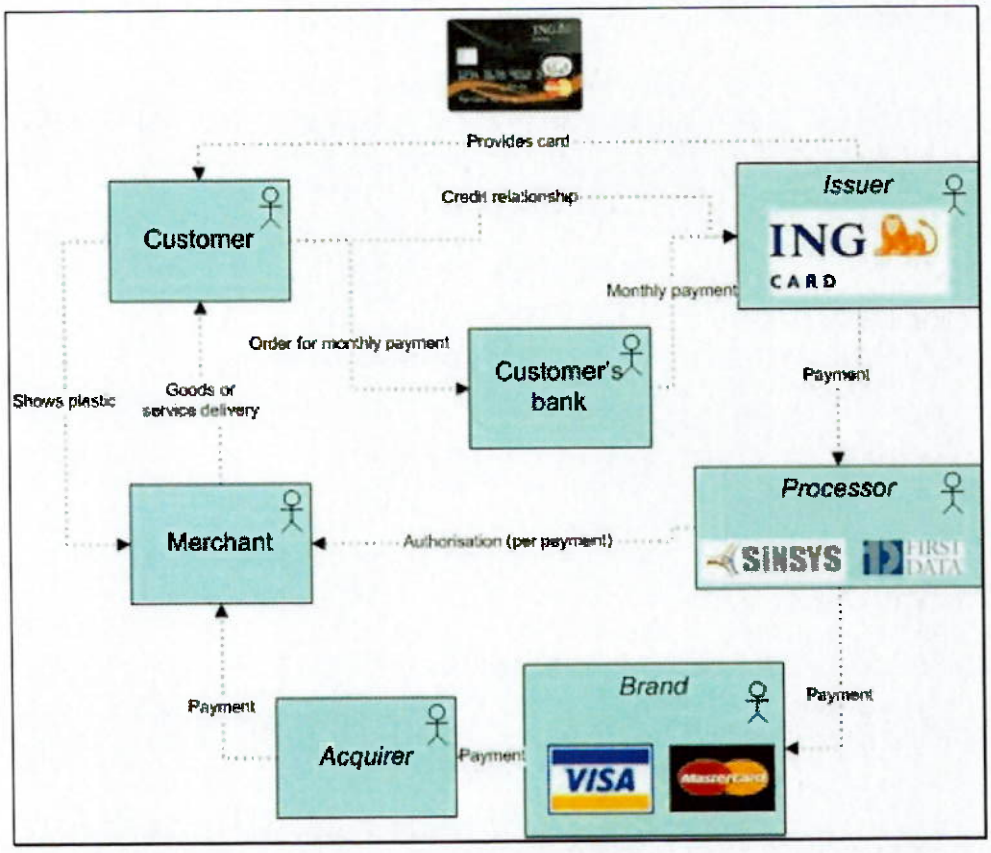


Figura 2.11 - Papéis no mercado de cartão de crédito mundial. (INGC, 2006)

[3] instituições (normalmente bancos)

Para começar, o ING Card teve alguns desafios particulares que significativamente influenciaram a escolha da solução de TI.

O primeiro desafio foi **manipular clientes** que já não tiveram uma conta atual do ING. Previamente, cartões de crédito somente eram fornecidos a clientes existentes com uma conta com a marca ING estabelecida, que era usado para pagamentos mensais. Agora, clientes novos podem ter uma conta atual em qualquer banco holandês. Todos os sistemas de clientes dentro do ING na Holanda foram identificados e foram ligados a uma conta cliente do ING. No entanto, estes sistemas não podiam ser usados pelo ING Card. Um novo sistema completo de gerenciamento de clientes foi necessário.

Em segundo lugar, vendas seriam concentradas **em canais diretos** (*internet*, central de chamados e mala direta) com instalações *online* de conta que forneceria ao cliente informação sobre suas transações numa base diária. Além disso, o cliente seria capaz de mudar as quantias de mensalidade (com restrições) e pedir um limite de crédito levantado por canais diretos.

Esta estratégia de mercado significava:

- Um *website* foi necessário para gerência *online* de conta pelo cliente;
- A central de chamados devia ser capaz de entrar aplicações de cartão e perguntas do cliente;
- Todas as funcionalidades de entradas de dados deveriam suportar todos os canais a fim de obter a mesma informação através de todos os canais; e
- Os processos de gerenciamento do cliente deveriam ser ligados ao processador de cartão, que cuida da distribuição do cartão, a conta real, e todas transações.

Logo veio o terceiro desafio, que foi relacionado a uma das competências de âmago de qualquer emissor: **pontuação de crédito**, que é

necessário para administrar o risco. O sistema deve avaliar todas aplicações de cartão usando calotes e indicadores de fraude.

O quarto desafio era em relação a desenvolvimentos futuros que foram conhecidos antecipadamente. O ING Card planejava um *roll-out* [4]através de **diversos países** envolvendo a entrega de produtos diferentes de cartão. O sistema necessitou ser suficientemente flexível para instalação em múltiplas situações geográficas, onde processos, produtos, e pontuação de crédito fossem passíveis de mudanças.

O desafio final surgiu porque o ING Card tornou-se comercialmente responsável por todas as pastas existentes de cartão de crédito de ING na Holanda (as marcas são Postbank e Banco ING), com mais que um milhão de clientes. Isto queria dizer que o novo sistema deveria se assentar em processos existentes das organizações. Logo tornou-se claro que a pontuação de crédito deveria ser executado centralmente e deveria estar disponível a todos os outros sistemas de cartão com o ING, assim estabelecendo uma única fonte para regras de negócio de crédito.

Em resumo, os seguintes requisitos tinham que ser satisfeitos:

Funcionais:

- administração de clientes;
- instalações *online* para o cliente ;
- suporte de multi-canal;
- pontuação de crédito;
- suporte para múltiplos países;
- suporte para múltiplos produtos.

Não-funcionais:

- agilidade para processos diferentes;
- agilidade para alterar características dos produtos;
- integração a outros processos/sistemas, baseados na mesma lógica de negócio.

[4] roll-out: implementação de um aplicativo ou sistema em outra localidade.

2.4.3. A Solução

Uma decisão foi feita na etapa inicial de construir em Websphere da IBM e plataformas da Oracle, solução a qual foi atribuído o nome de "Novo Sistema de Cartão" (NCS).

O nível necessário de agilidade finalmente foi alcançado por usar três princípios fundamentais de construção:

1. Uma arquitetura de camadas (conhecido como a "arquitetura de n-camadas").

2. A orientação-a-serviço (que eles também se referem a "o conceito de serviço").

3. Parametrização de classes selecionadas de negócio.

As três camadas tradicionais (apresentação, lógica de negócio e dados) foram enriquecidos por orientação-a-serviço pela apresentação de uma camada de serviço e uma camada de modelo de negócio, ambas residindo dentro da camada de lógica de negócio. Além disso, a fileira total de dados foi melhorada pela separação da camada de acesso aos dados da camada de armazenamento de dados. As camadas são retratadas na Figura 2.12 junto com as técnicas e as tecnologias usaram para sua implementação.

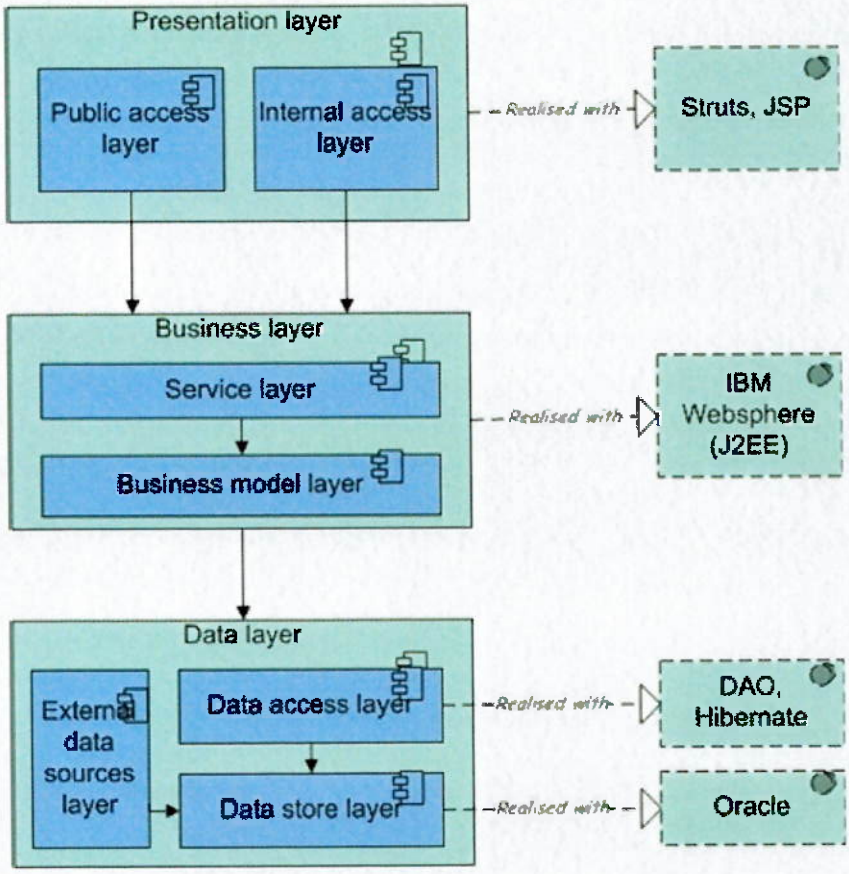


Figura 2.12 – As camadas da arquitetura NCS. (INGC, 2006)

A camada de apresentação (*presentation layer*) forma o exterior do sistema e consiste numa série de telas apoiando a automação dos processos principais de negócio. NCS atualmente tem os seguintes exemplos da camada de apresentação:

- Vários *web sites* públicos que fornecem as telas para solicitar um novo cartão. Estas telas chamam os serviços da camada de serviço que implementam o processo de aplicação.
- Um fechado e seguro ambiente Web que é acessível pelo cliente. Aqui o cliente pode se conectar, pode revisar transações e declarações mensais, e administrar sua conta. Fornece funcionalidades para solicitar um limite de crédito maior, mudando a quantia da mensalidade, solicitando um cartão secundário, e entrando mudanças pessoais (tal como ao endereço do cliente).

- Um sistema de entrada de dados para a central de chamados com funções semelhantes a esses que o cliente tem e com funções adicionais tal como ver o perfil do cliente e dados históricos.
- Um sistema de entrada de dados para o escritório de apoio, apoiando o processamento de aplicações de cartão.

A camada de modelo do negócio é acessível somente pela camada de serviço. Os serviços são funções que são significativas ao negócio. Os serviços não executam estas funções sozinhos, eles são como conchas ou fachadas, atrás da qual a função real é executada pela camada modelo de negócio.

Um exemplo de uma capacidade de serviço é *"findCreditRegistrationPerson"*, que cuida de achar a pessoa correta no sistema da agência de crédito (BKR é a agência de crédito nacional Holandesa). Veja a Tabela 2.2 para sua plena definição.

Name	findCreditRegistrationPerson
Service Group	NewAccountController
Release	2.0
Usage	Finds persons in the BKR system
Input	creditRegistrationPerson
Output	Collection of creditRegistrationPersons
Precondition	The creditRegistrationPerson must have agreed on checking BKR
Postcondition	NA
Actions	1. determine the necessary data 2. sends the data to BKR 3. receives the results from BKR 4. return the information
Business classes involved	creditRegistrationPerson

Tabela 2.2 – Um exemplo de definição de serviço (INGC, 2006)

A camada de apresentação podia dizer à camada de negócio, "Ache esta pessoa em BKR, aqui está os dados que você necessita para ele". A camada de apresentação então pode contar com a execução desta petição e em obter uma resposta. Este projeto tem três vantagens:

1. A camada de apresentação não necessita estar ciente de considerações técnicas (por exemplo, não necessita saber que conversão de alguns dados é necessitada antes de poder ser usado).

2. A seqüência em que serviços são chamados podem ser mudados facilmente sem qualquer mudança na camada de lógica de negócio.

3. O chamado de serviço não necessita ser mudado quando sua implementação é mudada, por exemplo, a necessidade de *"findCreditRegistrationPerson"* não é mudada quando a implementação da classe de *"CreditRegistrationPerson"* é mudada.

A independência de apresentação de lógica de negócio é aumentada em comparação com o modelo arquitetônico tradicional em três camadas. Por exemplo, a ordem de tarefas pode ser mudada ou outro *web site* pode ser adicionado com uma seleção limitada de serviços pelo cliente. Não há nenhuma necessidade modificar a camada de lógica de negócio para tais mudanças. Quando Cartão de ING é lançado em outro país, o serviço de *"findCreditRegistrationPerson"* terá outra implementação técnica para outra agência de crédito. O diálogo na camada de apresentação também não necessita ser mudado, porque o nome e chamado do serviço permanece o mesmo.

O terceiro princípio de construção de parametrização de classe foi aplicado dentro da camada modelo do negócio. Esta camada contem a lógica real de negócio, implementada como operadores em classes que representam o Modelo de Objeto de Negócio (BOM). O BOM determina a "linguagem" e consiste num conjunto de classes com seus inter-relacionamentos, implementados na forma de classes de Java, como se mostra na Figura 2.13.

Enquanto as classes “Contas”, “Terceiros” e “Cartão” são claros (eles apenas representam conta, cliente e cartão), as classes “Requisição”, “Regra” e “Produto” têm uma estrutura mais parametrizável que fornece uma flexibilidade maior. Quando uma variação de um produto é lançada numa campanha de marketing, só a configuração da necessidade da classe “Produto” é mudada, não a estrutura da classe. Isso é possível por definir um

atributo genérico chamado Característica *(Feature)* que pode conter qualquer possível característica para um produto e que faz a classe de Produto configurável como necessário.

Uma característica tal como "um ano livre de pagamentos" pode ser adicionado facilmente sem exigir redesenvolvimento. A implementação real de tal característica não é implementado em NCS mas é executado pelo processador de cartão; NCS só necessita remeter as características corretas ao processador. As classes "Requisição" e "Regras" são definidas num meio semelhante. Para a classe "Regra" há um dispositivo separado de regra que capacita a configuração flexível de regras de crédito.

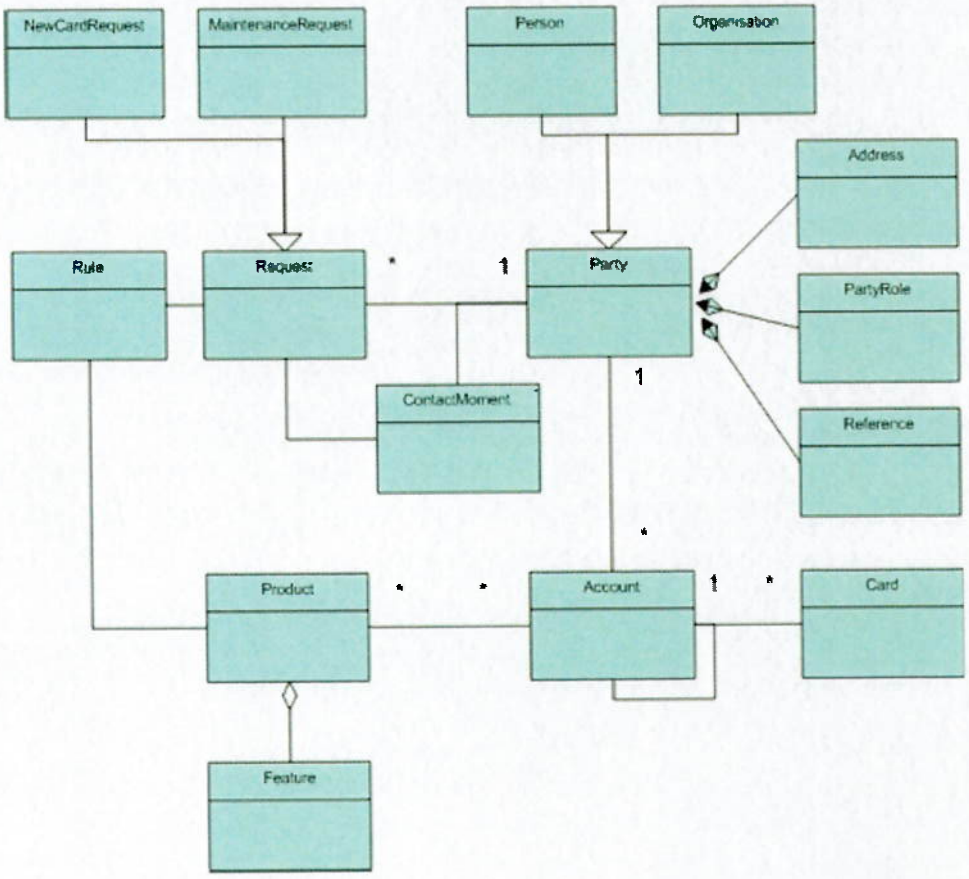


Figura 2.13 – Uma parte do Modelo de Negócio em notação de classes UML. (INGC, 2006)

Em resumo, a camada de negócio fornece serviços à camada de apresentação e cuida de transações na camada modelo de negócio de certa

maneira que reflete a linguagem da administração de cartão de crédito. A camada de modelo de negócio é montada com classes simples e parametrizáveis; as classes parametrizáveis capacitam mudanças em características de produto, regras de pontuação de crédito e itens de requisição sem impactar o sistema total.

2.4.4. Resultados

O NCS cumpriu expectativas por estabelecer como um sistema responsável para gerenciamento do cliente. Adicionalmente, o NCS executa todos os serviços que são oferecidos aos clientes na *Web* (tal como fazer mudanças de endereço ou solicitar um aumento do limite de crédito). Finalmente NCS também agora executa a avaliação de aplicações de cartão, assim permitindo mediar todos tipos de créditos e riscos de fraude. Os três princípios de construção de arquitetura em camadas, orientação-a-serviço e a parametrização de classe têm individual e coletivamente contribuído à realização dos requisitos originais, em particular esses associados com a agilidade (veja Tabela 2.3 abaixo).

realized in / *requirements*	Layered architecture	Service-orientation	Class parameterization
Online facilities			
Multi-channel			
Credit scoring			
Multi-product			
Multi-country			
Agility of processes			
Agility of product definitions			
Possibility to link to other processes			

Tabela 2.3 – Requisitos satisfeitos pelos princípios NCS . (INGC, 2006)

A **arquitetura em camadas** foi utilizada para entregar apoio de multi-canal. Se a mesma informação, (e parcialmente a mesma funcionalidade) é ser apresentado ao cliente via a Internet, o centro de chamado, ou por um empregado de escritório, então uma separação entre a camada de apresentação e camada de negócio é essencial para evitar a implementação duplicada da funcionalidade. A parametrização de classe foi utilizada para

entregar no tempo hábil para a apresentação de novos produtos e para adaptar a disponibilização de crédito e regras de descoberta de fraude.

A **orientação-a-serviço** tornou-se a mais importante de todos os três princípios. Por estabelecer uma fachada de lógica de negócio era possível realizar o *roll-out* NCS relativamente fácil em outros países. Além do mais, a orientação-a-serviço capacitou a otimização de processos sem mudar o sistema e a conectividade mesmo introduzindo outros processos e sistemas. A melhor demonstração do poder fornecido pela camada de serviço veio com o *roll-out* internacional. Está agora relativamente fácil para ING Card usar o NCS em qualquer outro país, mesmo quando há diferenças do ambiente importantes. Verificar a agência de crédito nacional, por exemplo, exige uma implementação diferente em cada situação. Há o trabalho de construção das interfaces restantes, mas o serviço que é chamado para executar a verificação não necessita mudar. Outro exemplo é a capacidade de um centro de chamado usar o próprio sistema preferido que apóia além do que somente o produto de cartão de crédito. Este sistema só necessita chamar os serviços apropriados do NCS quando um agente da central de chamados manipula uma requisição de cartão de crédito, usando as telas e diálogos da própria solução. Em outras palavras, não há nenhuma necessidade de uma integração profunda entre sistemas. Há muitas situações onde a independência da camada de serviço mostra seu valor, algum de que são ilustrados na Figura 2.14.

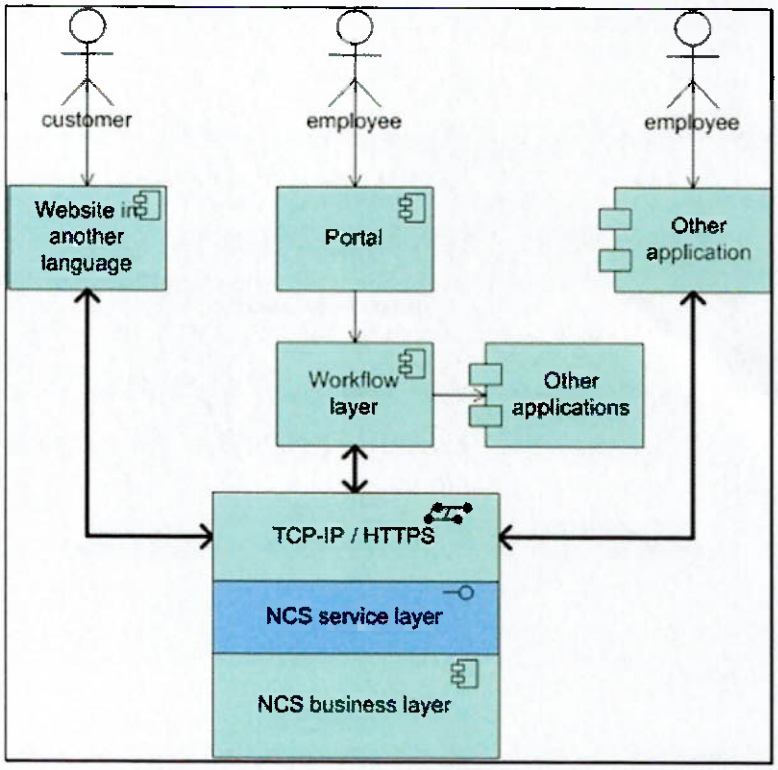


Figura 2.14 – Diferentes consumidores de serviço utilizando serviços em diferentes cenários de uso. (INGC, 2006)

Nem todos os aspectos de uma SOA conhecidos até então foram empregados no NCS. Por exemplo, as camadas de NCS comunicam via objetos J2EE, não mensagens. Portanto não há nenhum *bus de mensagem* e as tecnologias Web (SOAP, WSDL e UDDI) comumente usadas para desenvolver serviços Web são utilizadas. Os contratos de serviço foram publicados num documento Word e o governança destes serviços são processos manuais.

Na superfície, a solução não deve parecer muito diferente de aplicações API-tradicionais. Há uma diferença significativa, contudo. Os serviços não são vinculados com lógica específica. Em outras palavras, eles não fazem nada por conta própria porque sua responsabilidade primária é compor e invocar as funções apropriadas de negócio na seqüência correta. Estes serviços – os componentes formados por orientação-a-serviço – portanto, se estabelecem como membros muito mais independentes da

solução do que se tivessem sido desenvolvidas num projeto baseado em API tradicional.

A implementação de uma SOA do NCS foi projetada para ser realizada em fases. Na primeira fase, o NCS existe como um sistema simples com um número limitado de camadas de apresentação. Isto por que os serviços não necessitam ser publicados e podem ser invocados sem a utilização de protocolos uniformizados. Mas logo que os serviços do NCS necessitarem ser chamados de outras aplicações, tornar-se-á necessário invocá-los num meio uniformizado (bem possível a utilização de *Web services*) e fornecer mecanismos padrões para publicação e descoberta. Como um resultado, o NCS tem o potencial de participar de uma rede de múltiplos sistemas, que contribui a e nutre os níveis necessários de agilidade de negócio.

2.4.5. Lições aprendidas

Embora princípios arquitetônicos foram escolhidos e foram determinados antes do desenvolvimento do NCS, muitas versões exigiram novos princípios e abordagens que surgem durante o processo de desenvolvimento.

Abaixo estão listados algumas lições mais importantes que foram aprendidas:

- Gerentes de TI devem estar preparados para o novo papel de guardião do catálogo de serviços. SOA exige uma governança ativa da coleção ou inventário de serviços que é acumulado de modo que quando projetos subseqüentes entregam a duplicação acidental de serviços para uma nova funcionalidade , isto coerentemente é evitado.
- Todas as partes devem conhecer orientação a serviço. Isto inclui interessados de negócio assim como gerentes de aplicação e desenvolvedores de software. Quando qualquer um destas partes é deixada de fora, há um risco importante de equívoco e,

subseqüentemente, ocorrem pobres tomadas de decisão. Por exemplo, arquitetos de software tradicionalmente que só sabem sobre orientação-objeto podem ser inclinados a deixar os serviços juntos ou acoplados demais às funções reais de negócio.

- Os problemas com a granularidade de serviço (tal como a quantia de lógica encapsulada por cada serviço) deve ser resolvida durante a etapa de projeto e não de antemão. As conversas sobre granularidade podem tornar-se acaloradas e sem sentido se não relacionado à análise das funções reais que exigirão o encapsulamento.
- Deve ser uma regra que a camada de modelo do negócio só pode ser acessada via (e portanto somente abstraída pela) camada de serviço. Isto é uma tentativa de chamar operações da classe de negócio diretamente da camada de apresentação, especialmente quando o sistema ainda é isolado e não ainda conectado a outras aplicações nem para as camadas de apresentação. Desta perspectiva, a camada de serviço pode parecer redundante e pressões típicas e prazos apertados podem inclinar desenvolvedores a contornar os serviços como um atalho tático. Tais atalhos, no entanto, têm um efeito desastroso na entrega de soluções futuras que necessitará contar com o inter-conectividade uniformizada estabelecidas pelos serviços.
- O uso de ferramentas que forçam a consistência entre modelos (UML) é altamente recomendado. No projeto de desenvolvimento do NCS, usando uma única ferramenta para classe, atividade, e diagramas de seqüência provaram ser um "salvador de tempo" significativo.

3. MÉTODO PARA ANÁLISE, ESPECIFICAÇÃO, DESENVOLVIMENTO E GOVERNANÇA DE UMA SOA

3.1. Método Proposto

Iniciativas de implementação de uma arquitetura SOA podem potencialmente sofrer um ou mais dos seguintes problemas:

- Alta complexidade e curto espaço de tempo para lançar no mercado, conduzindo a um alto risco de fracasso.
- Baixa tolerância para fracasso por causa de despesas altas e dificuldade de quantificar o Retorno sobre o Investimento (ROI) associadas com o projeto de SOA.
- Requisitos dinâmicos e necessidades de negócio que necessitam um gerenciamento apropriado.

Desta forma, se faz necessário um método de software apropriado que forneça a sequência correta dos processos durante a fase inicial de planejamento SOA mas também seja adaptável e ágil para realizar as mudanças necessárias. Mas como sua empresa pode fazer a transição de um nível para outro enquanto ainda mantém um processo de desenvolvimento coerente ? Neste trabalho, propõe-se um método *best-of-breed* [5] para desenvolver e continuamente aperfeiçoar uma SOA.

3.2. O Método SOA-v360

Os principais fatores direcionadores de um típico projeto de desenvolvimento de software são o processo de desenvolvimento, o processo de gerenciamento e as tecnologias utilizadas. Um projeto genérico de desenvolvimento de software tem quatro variáveis: tempo, orçamento, escopo e qualidade. O comprometimento de uma das variáveis tem um impacto no projeto como um todo. Por causa das mudanças das necessidades do negócio, escopo e qualidade são dois dos mais difíceis

[5] *Best-of-breed*: comparativamente a melhor da categoria.

fatores para gerenciar. Complexidades tecnológicas podem conduzir a problemas com tempo e gerenciamento do orçamento também.

Projetos de SOA são significativamente mais complexos que projetos comuns de software, porque eles requerem um tempo maior, equipes multifuncionais que correspondentemente implicam em comunicações e logísticas mais complexas entre as equipes. Geralmente, uma arquitetura SOA não é bem definida, e a visão do resultado final freqüentemente não está clara no começo do projeto. Um projeto SOA pode ter um impacto muito positivo em uma organização, mas geralmente tem que seguir algumas regras as quais não se pode ignorar. Os fatores-chave que podem ajudar no sucesso da implementação são uma clara definição do processo de desenvolvimento, boas linhas de comunicação através das equipes do projeto que conhecem o negócio, e claras políticas de suporte e governança.

3.2.1. Visão Geral

SOA-v360 é um método *top-down* de análise, especificação, desenvolvimento e gerenciamento que busca ter uma visão de 360º de todas as fases do ciclo de vida SOA. Cada uma das fases , conforme Figura 3.1 , representa um distinto conjunto de atividades críticas para o sucesso da implementação SOA. Naturalmente, como em qualquer projeto, você pode e necessita adaptá-lo às características de sua organização.

Abaixo segue a descrição de cada perspectiva do método proposto SOA-v360. O método é agrupado em três perspectivas principais:

- **perspectiva Negócio**: se encontram as fases de modelagem do negócio e definição dos requisitos do negócio. Os principais envolvidos nas fases desta perspectiva são os executivos e os analistas de negócio. Os executivos são responsáveis pela definição das estratégias, objetivos do negócio e pela governança corporativa da organização que irá planejar, priorizar, gerenciar e medir os resultados dos planos do negócio, sendo que SOA é uma destas

iniciativas. Os analistas de negócio são responsáveis de definir os requisitos do negócio que serão fundamentais para a modelagem dos processos de negócio e dos serviços que irão compor a arquitetura SOA;

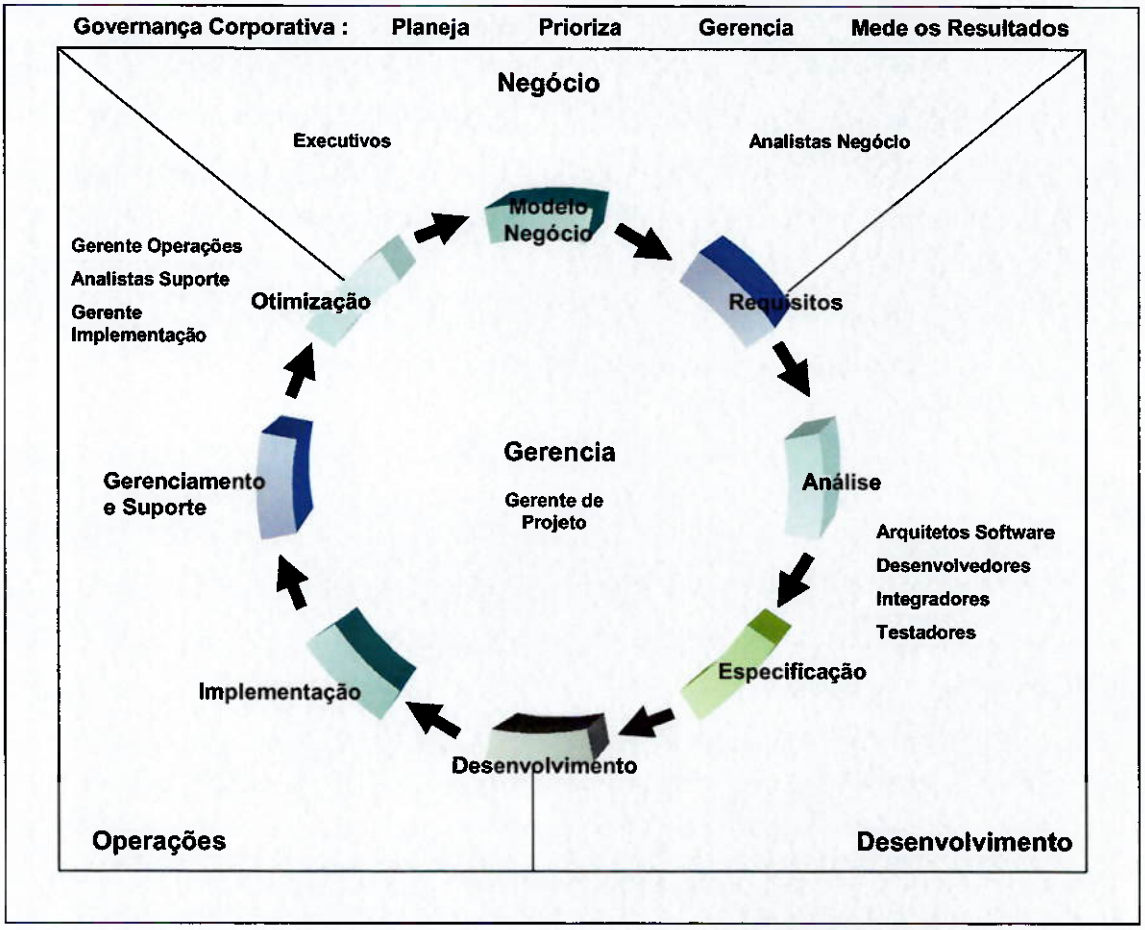


Figura 3.1 – As perspectivas, fases e responsáveis do Método SOA-v360

- **perspectiva Desenvolvimento**: se encontram as fases de análise, especificação e desenvolvimento. Os principais envolvidos nas fases desta perspectiva são os arquitetos de software, responsáveis pela análise orientada a serviço, pelos desenvolvedores e integradores que irão especificar e desenvolver os serviços, os integradores, responsáveis pela integração dos

serviços com outros sistemas e os testadores, responsáveis pela documentação e testes dos serviços.

- **perspectiva Operações**: se encontram as fases de implementação, gerenciamento e suporte e otimização. Os principais envolvidos nas fases desta perspectiva são os gerentes de operações do negócio, os analistas de suporte e os coordenadores de implementação, dependendo do porte do projeto. São os responsáveis para que as falhas nas fases de implementação sejam corrigidas e haja uma melhoria contínua e otimização da arquitetura.

No meio destas perspectivas se encontra o **gerenciamento do projeto**, cujo responsável é o gerente de projeto SOA, cuja responsabilidade é ter uma visão geral, ou uma visão 360° de todas as perspectivas, aplicando as técnicas de gerenciamento do projeto para que haja a integração de todas os responsáveis por cada uma das perspectivas.

3.2.2. Pré-requisitos do Método SOA-v360

3.2.2.1. Objetivos, Metas e Estratégias Corporativas

A principal razão de se escrever as metas e objetivos do negócio é procurar adequar e orientar o caminho a ser seguido para que a empresa esteja cumprindo sua missão em direção à sua visão. A principal diferença entre metas e objetivos é que a meta indica intenções gerais da empresa e o caminho básico para chegar ao destino que você deseja. Já os objetivos são as ações específicas mensuráveis que constituem os passos para se atingir a meta.

Nada mais importante que no estabelecimento dos objetivos da arquitetura corporativa de uma organização, que ele esteja alinhada ao planejamento estratégico corporativo.

3.2.2.2. Definir os requisitos do negócio

Quando os requisitos do negócio dirige todas as fases do ciclo de desenvolvimento e implementação SOA, o *gap* entre TI e negócio e diminuído consideravelmente. Assim, definir os requisitos do negócio será muito importante a fim de que a modelagem dos processos de negócio a arquitetura corporativa estejam alinhadas com as necessidades do negócio.

3.2.2.3. Modelar os processos de negócio

É muito importante determinar a necessidade de modelar, caso a organização não tenha claramente documentado os seus processos de negócio ou remodelar os seus processos caso já tenha documentado, antes de prosseguir nas demais fases do método. Ferramentas como BPM ou outras fornecidas pelos *players* podem ser utilizadas, como mostra a Figura 3.2 abaixo:

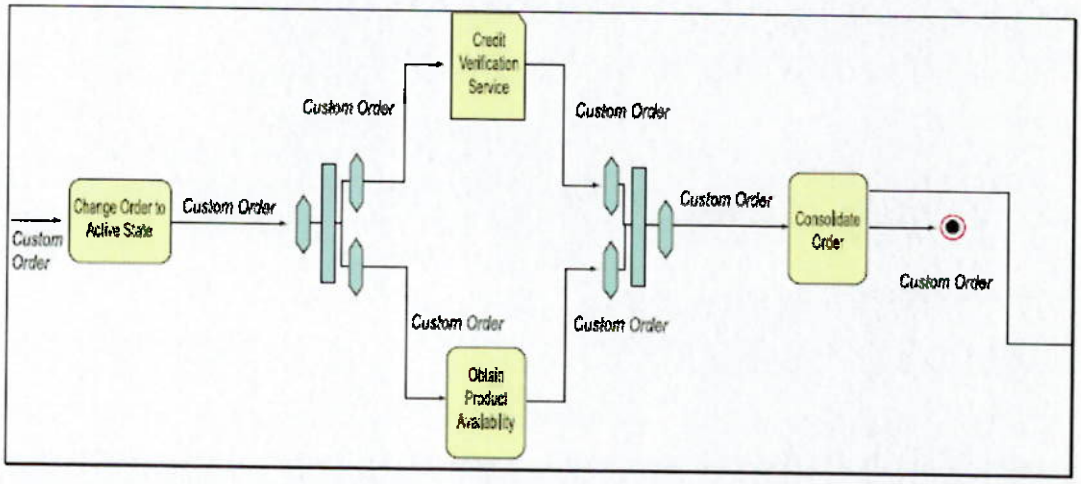


Figura 3.2 - Exemplo de modelagem de processos de negócios (BROWN & JOHNSTON ,2005)

3.2.3. Planejamento e Governança SOA

SOA tem um grande potencial para promover a eficiência dos processos de TI em uma organização. Mas implementá-la em uma

organização, é necessário muito mais do conhecimento técnico: conhecimentos e habilidades em gerenciamento de projeto e pessoas também são fundamentais. Os princípios em governança de TI podem auxiliar nestes aspectos.

A Governança fornece uma estrutura a fim de suportar os objetivos de negócio nos níveis estratégico, funcional e operacional. Ela define as regras, processos, métricas e modelos organizacionais necessários para que sejam efetivos os processos de planejamento, tomada de decisão e controle da arquitetura SOA para atingir as necessidades do negócio e os objetivos propostos. Assim , os modelos de governança estabelecem:

- O quê fazer ;
- Como fazer ;
- Quem deve fazer:
- Como deve ser mensurado.

Nesta fase, você estabelece a necessidades de projeto SOA em sua organização. Os principais *artefatos* a serem entregues nesta fase são:

- Visão SOA e escopo do projeto.
- Estratégia SOA .
- Análise do ROI .
- Planos de comunicação.
- Modelos de suporte e governança

Os clientes nem sempre compreendem os benefícios de novos produtos de software em seus negócios. As equipes que definem a estratégia SOA necessitam influenciar as equipes do projeto para ajudá-los a identificar os problemas no negócio.

Analistas inter-funcionais e gerentes de projeto analisam os negócios dos clientes para determinar as vantagens de uma solução SOA. Os fatores chave que os analistas examinam são as operações internas dos clientes, interações com seus sócios, fornecedores, e consumidores e seu modelo completo de negócio. Isto auxilia os analistas a desenvolverem e recomendarem uma estratégia de SOA.

Uma análise de ROI deve mostrar claramente os benefícios financeiros à organização -- no médio, curto e longo prazos. Um artefato bem importante também nesta fase é o plano de comunicações. Finalmente, as equipes de TI do projeto e os interessados do negócio sabem o negócio melhor que as equipes de arquitetura e analistas. O plano de comunicações deve assegurar que estes interessados internos entendam e estejam envolvidos no processo.

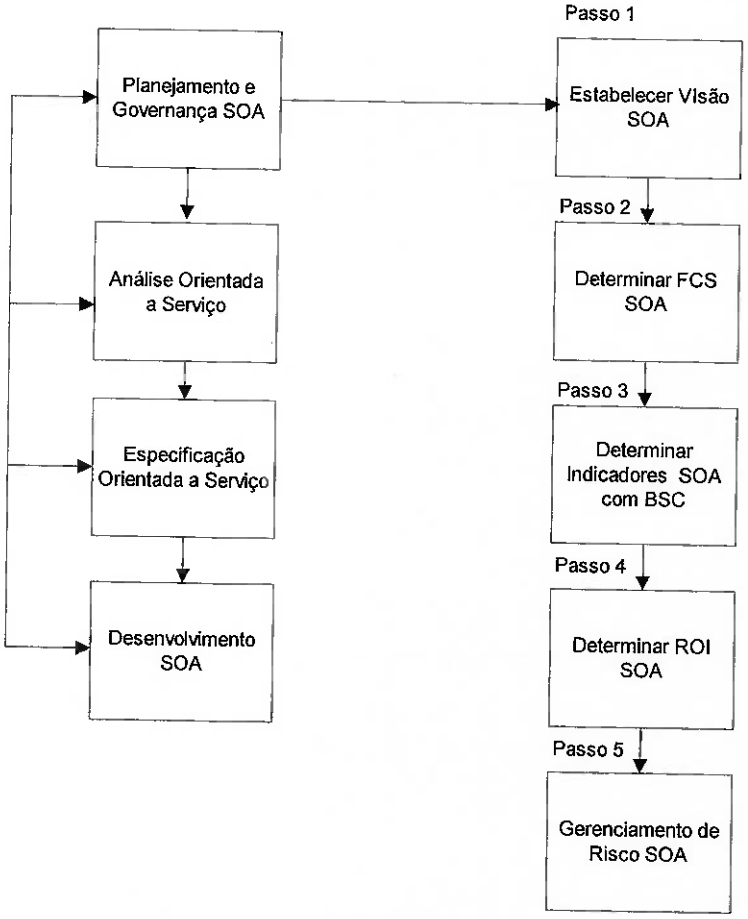


Figura 3.3 – Os passos do processo de Planejamento e Governança SOA

3.2.3.1. Passo 1: Estabelecer uma Visão SOA

O segundo passo para criar uma arquitetura SOA é estabelecer uma clara visão do que o SOA será, quais os limites ou fronteiras que ela atingirá na organização, bem como quais benefícios (tangíveis e intangíveis) seremos capazes de obter com sua implementação. Freqüentemente, as

companhias insistem em implementar uma arquitetura SOA sem claramente identificar o valor obtido ou o resultado ideal a ser alcançado para o negócio com sua implementação.

3.2.3.2. Passo 2: Determinar os Fatores Críticos de Sucesso SOA

Os Fatores Críticos de Sucesso (FCS) são as habilidades e os recursos que a empresa ou algum projeto precisa necessariamente ter para atingir os objetivos propostos. Limite sua lista de FCS a não mais do que 4 ou 5.

3.2.3.3. Passo 3: Criar Indicadores SOA com BS (Balanced Scorecard)

A visão SOA estabelece uma série de benefícios ao negócio com a implementação SOA. Estas promessas são certamente atrativas , porém mais importante que isso, é estabelecer os valores mensuráveis relacionando cada benefício a valores monetários, valores que serão utilizados a seguir para cálculo do ROI (Retorno do Investimento) em SOA.

O *balanced scorecard* de Kaplan e Norton conforme já explicado na revisão de conceitos, decompõe a estratégia em objetivos, indicadores, metas e iniciativas de uma forma lógica, baseada em relações de causa e efeito, vetores de desempenho e relação com os fatores financeiros . Deve ser decomposta em quatro dimensões do negócio: financeira, clientes, processos internos e aprendizado e crescimento. Tais perspectivas respondem a quatro perguntas:

- Para ter sucesso financeiro no projeto SOA, como devemos nos mostrar aos nossos acionistas?
- Para atingir nossa visão SOA, como devemos nos mostrar aos nossos clientes?
- Para satisfazer nossos acionistas e clientes, em quais processos devemos ser excelentes?

- Para atingir nossa visão SOA, como manteremos nossa habilidade de mudar e melhorar?

Desta forma, a arquitetura SOA passa a ser vista como uma estratégia ("para onde iremos") e a estratégia ("como iremos") dentro da organização, tendo sua própria visão com objetivos, indicadores e metas específicos, mas é muito importante que todas estas variáveis estejam alinhadas com o planejamento estratégico da organização.

Como exemplo da aplicação das perspectivas BSC em uma estratégia SOA, poderíamos ter o seguinte quadro o que vai compor o **Mapa Estratégico SOA** para o tema de diminuição do custos operacionais de TI da empresa através da reutilização dos ativos de software:

Tema: Diminuir o custos operacionais de TI da empresa pela reutilização dos ativos de software					
Perspectivas	**Mapa Estratégico**	**Balanced Scorecard**		**Plano de Ação**	
		Indicador	**Meta**	**Iniciativa**	**Verba**
Finanças	- Diminuir custos de TI da empresa	- Composição do custo	- 10%	- Planejamento financeiro	$ 000,
Clientes	- Contratos de baixo custo e com bom retorno para o cliente	- Satisfação do cliente (tempo de atendimento, número de reclamações)	100%	- Pesquisa de satisfação do cliente interno e parceiros de negócio	$000,
Processos	- Produtividade dos recursos (envolvidos nos projetos) ; - Redução dos custos unitários ; - Redução das despesas operacionais	- Razão horas alocadas por total de horas previstas - Custo unitário por projeto	Menor ou igual a 80% da Receita	- Reengenharia de processos	$000,

Tema: Diminuir o custos operacionais de TI da empresa pela reutilização dos ativos de software					
Perspectivas	Mapa Estratégico	Balanced Scorecard		Plano de Ação	
		Indicador	Meta	Iniciativa	Verba
Aprendizado e Crescimento	- Treinamento em tecnologia web services para técnicos de TI e gerenciamento de projetos para gerentes de projeto	- Número de gerentes e técnicos treinados	100%	- Programação de treinamento junto à área de recursos humanos.	$000,

Tabela 3.1 – Exemplo da composição de um mapa estratégico SOA

Observa-se que para cada plano de ação do mapa estratégico está relacionada uma verba a qual é o custo relacionado para implantar aquela iniciativa correspondente. Para cada um dos temas estratégicos SOA identificados pela organização, geralmente temas que estão alinhados com os benefícios SOA, deve ser desenvolvido um mapa estratégico respectivo a fim de que o ROI da arquitetura SOA seja mais fácil de ser mensurado e calculado.

3.2.3.4. Passo 4: Determinar o ROI do SOA

Enquanto o mundo é limite e determina a construção de uma arquitetura SOA dentro de sua empresa -- talvez mesmo entre empresas -- poucos tentam determinar seu Retorno em Investimento (ROI) para justificar esta implementação. Atualmente, arquitetos e pessoas de negócio devem trabalhar conjuntamente para determinar se as mudanças propostas são adequados para um negócio. De fato, a aplicação de um SOA tem graus diferentes de ROI, dependendo do domínio de problema. O custo de implementação de um SOA diretamente deve estar relacionado aos benefícios ao negócio.

Implementamos SOA por duas razões importantes. Primeiro é a capacidade de economizar no desenvolvimento pela ***reutilização de serviços***. Estes serviços podem ter sido construídos dentro ou fora da companhia, e quanto mais serviços forem reutilizáveis de sistema para sistema, maior o ROI do SOA. O segundo é ***a capacidade de mudar*** a infra-estrutura de TI mais rápido para se adaptar às necessidades variáveis do negócio. Isto, naturalmente, fornece uma vantagem estratégica enorme e assim permite para o negócio possa ter melhores possibilidades de sobrevivência a longo prazo. Enquanto determinar o ROI em agilidade é mais difícil de converter em termos financeiros, nós sabemos que o benefício está presente e é muito importante.

Reuso de Serviços

Sob o conceito do reuso do serviço, temos alguns pontos para determinar para melhor definir o valor. São os seguintes:

- o número de serviços que são reusáveis ;
- a complexidade dos serviços
- o grau de reuso de sistema para sistema.

O número de serviços reutilizáveis é o número real de novos serviços criados, ou, existindo serviços abstratos, estes são potencialmente reutilizáveis de sistema para sistema. A complexidade dos serviços é o número de funções ou pontos de objeto que compõem o serviço. Usamos funções tradicionais ou pontos de objeto como um meio comum de expressar a complexidade em termos dos tipos de comportamentos que o serviço oferece. Finalmente, o grau de reutilização de sistema para sistema é o número de vezes que você realmente reutiliza os serviços. Podemos considerar este número como uma porcentagem.

Agilidade

A agilidade é uma vantagem estratégica que é difícil de medir em termos monetários, mas não impossível. Nós primeiramente necessitamos determinar algumas coisas sobre o negócio, incluindo:

- O grau de mudança no tempo ;
- A capacidade de adaptação à mudança ;
- O valor relativo da mudança.

O grau de mudança no tempo é realmente o número de vezes sobre um período determinado que o negócio se reinventa para se adaptar a um mercado. Assim, enquanto uma companhia de produção de papel pode ter um grau de mudança de 5 por cento sobre um período de 5 anos, uma companhia de alta de tecnologia pode ter uma 80 por cento de mudança sobre o mesmo período.

A capacidade de adaptar às mudanças é um número que declara que a capacidade da companhia reagir à necessidade mudar no tempo. A noção é que o uso de um SOA fornece uma melhor capacidade de mudar para se ajustar às mudanças necessárias no negócio.

Finalmente, os valores relativos de mudança é a quantia de dinheiro feito como um resultado direto de mudar o negócio. Por exemplo, uma organização varejista, a capacidade de estabelecer um programa freqüente de compras para reagir a expectativas variáveis de mercado, e para os aumentos resultantes em venda que a mudança provocou.

Determinar o ROI do SOA não é uma ciência exata, mas com alguma análise e alguns pontos realistas de dados, pode-se compreender quanto valor sua implementação de SOA trouxe ou não. Outra vez, nós necessitamos saber dos custos para se justificar o uso desta tecnologia, e a informação apresentada aqui deve ajudá-lo ao longo do caminho para criar o próprio case de negócio.

3.2.3.5. Passo 5: Gerenciamento de Risco SOA

Sempre que uma organização se engaja em uma nova iniciativa, diversos elementos fundamentais devem estar presentes para garantir o sucesso desse esforço. Com um projeto SOA não é diferente. Há componentes que devem estar presentes ao iniciar a implementação de um

processo bem-sucedido de gerenciamento de riscos do projeto, e que devem permanecer presentes durante esse processo. São eles:

- **O patrocínio executivo**: os administradores seniores devem oferecer um suporte inequívoco e entusiasta ao processo de gerenciamento de riscos do projeto. Sem esse suporte, os interessados nessa iniciativa podem vir a resistir ou a se opor aos esforços de implementação do gerenciamento de riscos para tornar mais segura a organização.

- **Um lista bem definida dos interessados no gerenciamento de riscos**: a Equipe de gerenciamento de riscos necessita conhecer todos os interessados, inclusive os membros da própria equipe e os patrocinadores executivos. Isso inclui ainda os indivíduos responsáveis pelos ativos comerciais a serem avaliados. A equipe de TI responsável pela criação, implementação e gerenciamento dos ativos comerciais também faz parte do grupo de interessados.

- **Os interessados devem ser identificados** de forma que possam participar no processo de gerenciamento de riscos. A Equipe de gerenciamento de riscos deve dedicar algum tempo para auxiliar essas pessoas a compreender o processo e o modo em que ele as ajudará a proteger seus ativos e economizar dinheiro a longo prazo.

- **Maturidade corporativa** em relação ao gerenciamento de riscos: mesmo em uma organização com processos informais, como esforços aleatórios iniciados em resposta a problemas específicos, a implementação do processo poderá parecer um desafio grande demais. Contudo, ele poderá se mostrar eficaz em organizações mais maduras em termos de gerenciamento de riscos. A maturidade fica evidente através de aspectos como processo de desenvolvimento bem definidos e a compreensão e aceitação profundas do gerenciamento de riscos em diversos níveis corporativos.

- **Uma atmosfera de comunicações abertas**: Diversas organizações e projetos operam de acordo com a necessidade de conhecimento momentânea, o que, em geral, causa mal-entendidos e prejudica a capacidade de uma equipe de criar uma solução bem-sucedida. O processo de gerenciamento de riscos SOA exige uma abordagem de comunicações abertas e honestas, tanto no âmbito da equipe como dos interessados. Um fluxo contínuo de informações não só reduz os riscos de mal-entendidos e o desperdício de esforços como também assegura que todos os membros da equipe possam contribuir para a redução de incertezas que afetem o projeto. A discussão aberta e honesta sobre os riscos identificados e os controles que podem ser usados para atenuar esses riscos é essencial para o sucesso do processo.

Podemos definir o gerenciamento de riscos como um processo contínuo com quatro fases principais:

1. ***Avaliando os riscos*** — Identificação e priorização de riscos aos negócios.
    - Planejar a coleta de dados — Discuta os fundamentos do sucesso e a orientação da preparação.
    - Coletar dados sobre riscos — Estruture o processo de coleta e análise de dados.
    - Priorizar os riscos — Estruture as etapas necessárias para qualificar e quantificar os riscos.
2. ***Oferecendo suporte as decisões*** — Identificação e avaliação das soluções de controle de acordo com um processo definido de análise de custo/benefício.
    - Definir os requisitos funcionais — Defina os requisitos funcionais para atenuar os riscos.
    - Selecionar soluções de controle possíveis — Estruture a abordagem para identificar as soluções de atenuação.

- Rever a solução — Avalie os controles propostos em relação aos requisitos.

- Estimar a redução de riscos — Tente compreender a redução na exposição ou probabilidade de riscos.

- Estimar o custo da solução — Avalie os custos diretos e indiretos associados às soluções de atenuação.

- Selecionar a estratégia de atenuação — Faça a análise de custo/benefício para identificar a solução de atenuação mais econômica.

3. ***Implementando controles*** — Implementação e operação de soluções de controle para reduzir os riscos aos negócios.

   - Buscar uma abordagem holística — Incorpore pessoas, processos e tecnologia na solução de atenuação.

   - Organizar por defesa em profundidade — Organize soluções de atenuação para a empresa toda.

4. ***Analisando a eficácia do programa*** — Análise do processo de gerenciamento de riscos quanto à sua eficácia, e verificação dos controles quanto ao grau de proteção esperado.

   - Analisar a eficácia do programa — Avalie o programa de gerenciamento de riscos para determinar oportunidades de melhoria.

3.2.4. Análise, Especificação e Desenvolvimento SOA

Uma vez que o planejamento da arquitetura tenha sido realizado e que as diretrizes que orientarão as decisões e gerenciamento do ciclo de vida da arquitetura SOA estejam tomadas, as fase de análise, especificação e desenvolvimento podem ter início.

3.2.4.1. Análise Orientada a Serviço

Esta fase é de longe a mais crítica do projeto SOA, onde será definido quais são e como devem ser os serviços. O negócio e envolvimento de equipe de projeto aqui finalmente determina o êxito do projeto.

Longe de ser uma análise definitiva, esta análise passa a ser uma análise complementar que devem se juntar à outras existentes na organização. Como vimos na revisão de conceitos, a definição e coreografia dos serviços estão estritamente relacionadas aos processos de negócio.
Esta fase do ciclo de vida do projeto envolve atividades tais como, também detalhada na Figura 3.4 abaixo:

- Definição do escopo e dos aplicativos que irão compor o SOA
- Identificação e modelagem dos serviços
- Definição de requisitos não-funcionais
- Definição e documentação da arquitetura SOA

Os principais *artefatos* desta fase são:

- Escopo da análise
- Modelo de serviços
- Casos de uso e realização dos casos de uso
- Requisitos não-funcionais
- Arquitetura SOA

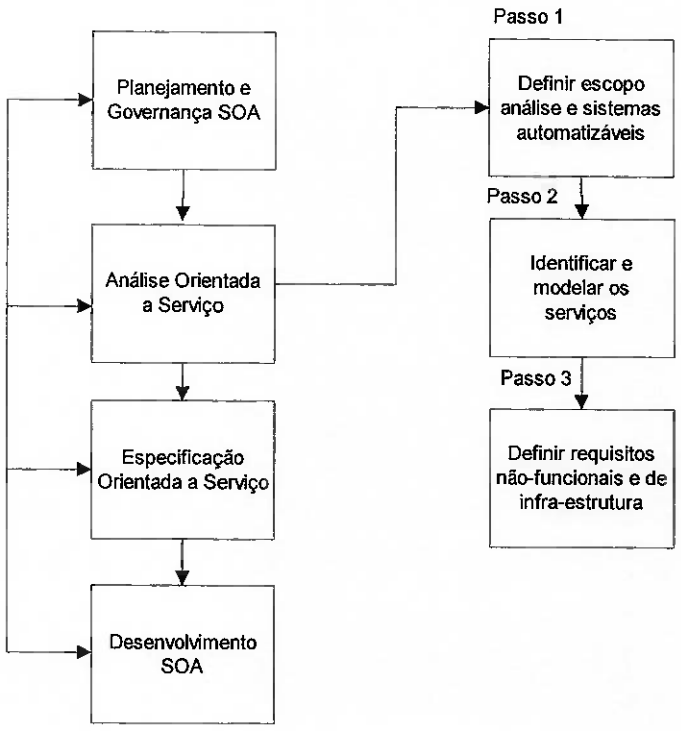


Figura 3.4 – Os passos do processo de Análise Orientada a Serviço

3.2.4.1.1. Passo 1: Definir escopo da análise e sistemas automatizáveis

Inicialmente devem ser considerados, dependendo do porte da organização, quais unidades de negócio devem ser consideradas no contexto da análise, bem como quais sistemas aplicativos são passíveis de automatização. De acordo com o modelo de negócio, pode-se saber se um sistema é crítico ou não para uma organização e deve ou não ser automatizado.

3.2.4.1.2. Passo 2: Identificar e modelar os serviços

Esta fase é um processo que consiste em agrupar as operações de serviços em um contexto lógico. Estes grupos acabam formando os candidatos à serviços que mais tarde irão compor um menu de serviços. Algumas questões iniciais que podemos considerar nesta fase são:

- Quais serviços são críticos para o negócio e necessitam ser desenvolvidos ?
- Quais lógicas devem ser encapsuladas para cada serviço ?

Para responder à estas questões preliminares são necessárias algumas atividades como:

- Definir um conjunto inicial de candidatos às operações de serviços.
- Agrupar as operações de serviços candidatas dentro de contextos lógicos. Estes contextos representam os candidatos à serviço.
- Identificar o reuso potencial das lógicas encapsuladas.

Para que estas atividades sejam realizadas os seguintes atividades principais são importantes:

(a) Decompor os processos de negócio

Como foi visto na revisão de conceitos, o escopo de um *business service* pode variar, como por exemplo, um serviço pode representar apenas um passo dentro de um processo de negócio, uma parte de um sub-processo dentro de um processo maior , ou até mesmo um processo inteiro. Desta forma, se torna necessário verificar a granularidade dos processos de negócio documentados e, caso não estejam no nível mais detalhado possível, este detalhamento será necessário.

(b) Excluir passos do processo não apropriados para encapsulamento do serviço

Algumas atividades dentro dos processos de negócio podem ser facilmente identificadas como não pertencente à lógica que deve ser encapsulada por um serviço candidato. Exemplo incluem processos manuais ou atividades executadas por lógicas de sistemas legados para as quais não são pertinentes a seu encapsulamento nos serviços.

(c) Identificar os casos de uso para os candidatos à serviços

Para cada candidato à serviço é necessário definir os casos de uso, detalhando a sequência lógica , como por exemplo, no processamento de uma PO (Ordem de Compra) de serviço teríamos a seguinte realização do caso de uso:

1. Recebe a Ordem de Compra.
2. Validação da OC.
3. Se o documento é inválido, então rejeita o documento e envia notificação de rejeição e finaliza processo.
4. Transforma o documento em formato XML num formato para ser gravada no sistema nativo, ERP, por exemplo.

Abaixo na Figura 3.5 segue um exemplo da aplicação da realização do casos de uso através de um diagrama de um pedido de compra:

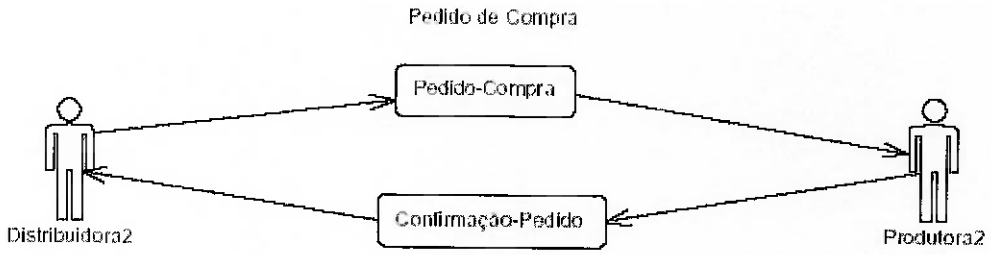


Figura 3.5 - Exemplo da utilização do diagrama de caso de uso para um pedido de compra

(d) Aplicar os princípios de orientação aos serviços candidatos

Até agora as atividades foram agrupadas dentro dos processos de negócio existentes. Para que os serviços verdadeiramente sejam serviços SOA, é necessário realizar alguns ajustes aplicando os princípios de orientação à serviço. Um dos princípios SOA têm especial significância durante a fase de modelagem é a **reusabilidade** .No exemplo anterior, pode-se identificar uma dependência direta entre os passos 2 (Validação OC) e 3 (Se o documento é inválido, envia notificação) . Aplicando o princípio da reusabilidade, pode-se combinar os dois dentro de um único serviço "Valida OC e envia notificação, se requerida".

3.2.4.1.3. Passo 3: Definir requisitos não-funcionais

O levantamento correto e preciso dos requisitos não-funcionais é extremamente importante em qualquer tipo de aplicação.

Os requisitos podem ser divididos em duas categorias:

- Funcionais: especificam as funções realizadas pelo produto.
- Não-Funcionais: requisitos de desempenho e outros atributos de qualidade, assim como requisitos lógicos dos dados e restrições.

Dentre os requisitos não-funcionais de uma aplicação, temos:

- **Tempo de resposta**: tempo total entre o momento em que o cliente submeteu a requisição até o momento em que a resposta chegou completamente até o cliente.
- **Vazão**: número de requisições executadas por unidade de tempo.
- **Disponibilidade**: fração de tempo em que o *site* está operacional. Um *site* pode permanecer indisponível devido à manutenções programadas ou falhas inesperadas.
- **Escalabilidade**: este conceito está ligado às mudanças de desempenho provocadas pela adição de recursos. A adição de novos recursos pode se dar de duas formas:aumento do número de servidores (escalabilidade horizontal) ou incremento do poder de servidores existentes , através do aumento de memória, por exemplo (escalabilidade vertical).
- **Segurança**: o nível de segurança de acesso desejado tem profundo impacto na construção do produto e normalmente restringe as opções de desenhos existentes.

3.2.4.2. Especificação Orientada a Serviço

Na fase de Análise Orientada a Serviço, foram determinados quais tipos de serviço você necessita e os usos de caso e realização dos usos de caso foram documentados. Os serviços com atividades ou operações similares foram agrupados.

Neste fase, os arquitetos de SOA (geralmente um subconjunto dos arquitetos da empresa) necessitam envolver arquitetos do projeto para certificar-se o projeto apresentado pela equipe de SOA funcionarão para outros projetos específicos.

Algumas questões chave devem ser respondidas nesta fase:

- Como as definições físicas de interface de serviço podem ser derivadas dos serviços modelados durante a fase de análise orientada à serviço ?
- Quais características SOA devem ser empregadas e suportadas ?
- Quais padrões da indústria serão requeridos pelo SOA para implementar a arquitetura ?

As atividades principais desta fase são as seguintes, conforme mostra a Figura 3.6 abaixo:

- Identificar as tecnologias e plataformas de desenvolvimento
- Definir as especificações de interface.
- Identificar as composições de serviço potenciais.

Os principais artefatos desta fase são:

- Documento com especificação detalhada
- Modelo de programação da aplicação

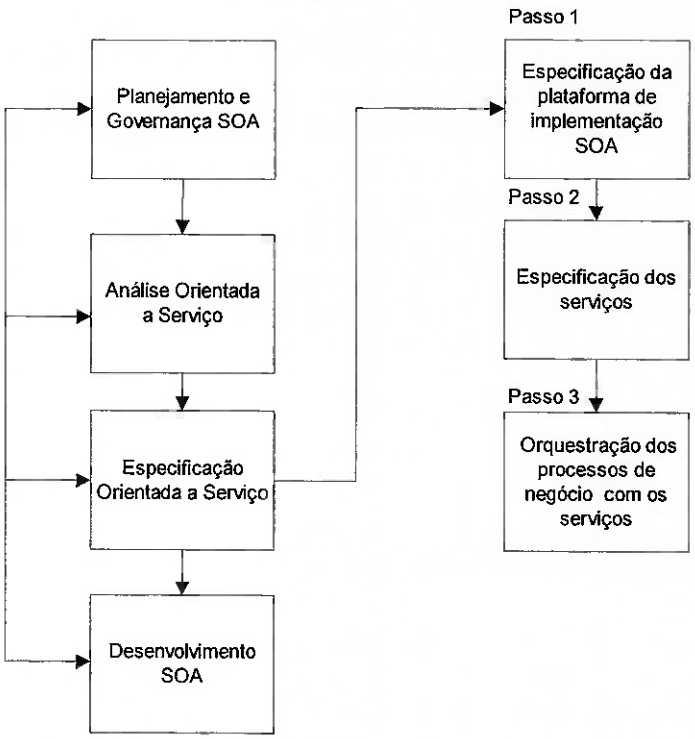


Figura 3.6 – Os passos do processo de Especificação Orientada a Serviço

3.2.4.2.1. Passo 1: Especificação da tecnologia e escolha da plataforma de desenvolvimento SOA

Antes, para que os serviços sejam desenvolvidos, a arquitetura necessita estar bem definida. Nesta atividade, se define um conjunto padrão de camadas de serviço e então a escolha de tecnologias abertas para suportar aquelas camadas como Web services, por exemplo.

As principais e mais conhecidas plataformas de desenvolvimento existentes atualmente no mercado são Java da *Sun Microsystems* e .NET da *Microsoft*. As plataformas .NET e J2EE possuem o foco para o mercado de aplicações corporativas e para a tecnologia Web Services. A seguir são apresentadas algumas diferenças entre as plataformas:

- J2EE é centrado na linguagem Java e é multiplataforma ; .NET é centrado na plataforma Windows mas você pode usar diversas linguagens (VB.NET , C# , J# , Cobol ,etc...).

- Todos os componentes que são distribuídos no framework J2EE (EJB , Servlets) são escritos em Java .
- O framework .NET permite o desenvolvimento em qualquer linguagem que for suportado pelas ferramentas Microsoft.
- No J2EE você esta restrito à linguagem Java ; no .NET você está restrito à plataforma Windows.

Estratégia

J2EE é basicamente uma série de padrões. .NET é um produto da estratégia Microsoft baseado na evolução do Visual Studio 6.0. Por trás do padrão J2EE a Sun procurou reunir as maiores empresas de software afim de adaptar a interface J2EE, como BEA Systems, IBM e Oracle. .NET é um esforço quase que isolado da Microsoft para atingir o mercado de *Web Services*.

Compatibilidade e Legado

É muito fácil a interligação com o código legado através da nova arquitetura JCA - *Java Connector Architeture*. .NET também oferece uma boa integração com o legado seu Server 2000 mas possui limitações de conectividade para selecionar sistemas. Quanto à compatibilidade , a atualização do código VB 6.0 para VB.NET , apresenta sérios obstáculos devido a introdução dos novos conceitos de orientação a objeto ao VB.NET.

Portabilidade

O Java esta disponível para qualquer plataforma - Win32 , Unix , Mainframe , o que facilita muito a portabilidade das aplicações J2EE. Quanto a plataforma .NET existe um esforço da Microsoft em oferecer meios para que as aplicações .NET rodem em outras plataforma que não o Windows.

3.2.4.2.2. Passo 2.1: Especificação dos serviços – entidades

Serviços de negócio modelados focando as entidades significa que as entidades modeladas em UML através dos processos de negócio podem

ser utilizadas para especificar os serviços. Por exemplo, o serviço reserva de veículo tem a entidade "Reserva" , conforme Figura 3.7 abaixo e tem associada a ela as seguintes operações e atributos:

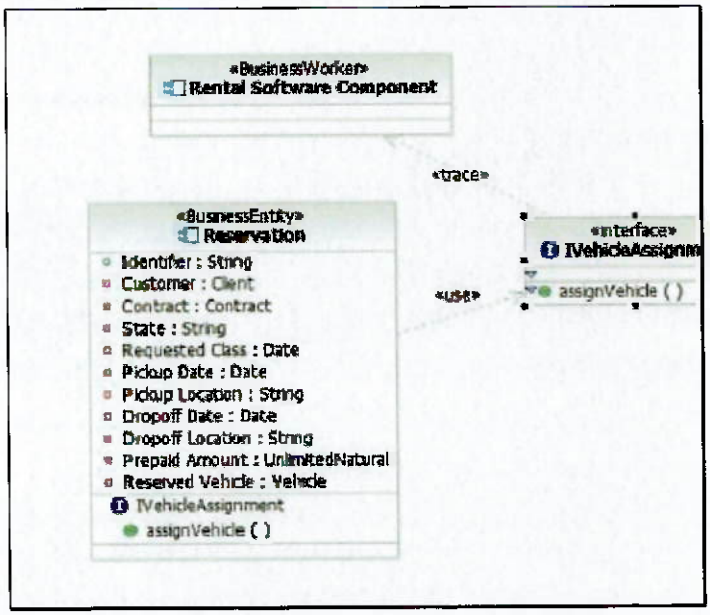


Figura 3.7 – Exemplo de especificação de uma entidade de serviço utilizando notação UML (BROWN & JOHNSTON ,2005)

3.2.4.2.3. Passo 2.2: Especificação dos serviços - execução

Nesta fase é necessário realizar o seqüenciamento dos serviços, podendo-se utilizar a notação da linguagem UML para o diagrama de sequência. **Diagrama de sequência** é o tipo de diagrama usado em UML (*Unified Modeling Language*), representado a sequência de processos (mais especificamente, de mensagens passadas entre objetos) e também pode ser utilizada para representar o seqüenciamento dos serviços, conforme mostra a Figura 3.8 abaixo:

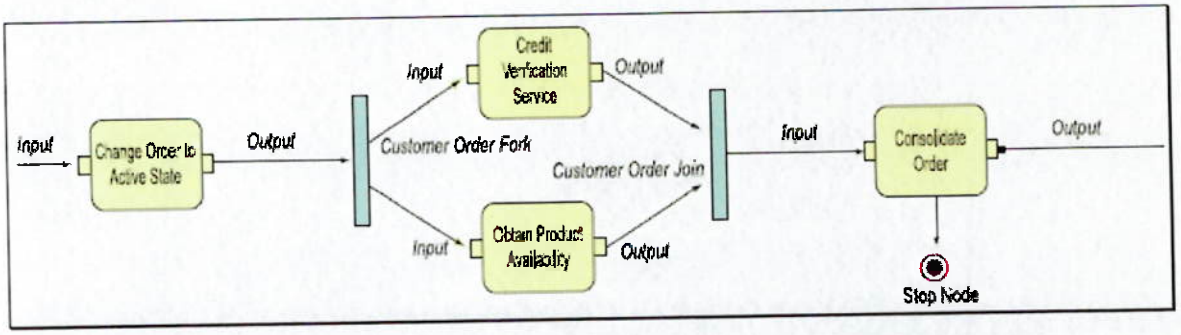


Figura 3.8 - Exemplo de um diagrama de sequência de serviços utilizando notação UML (BROWN & JOHNSTON ,2005)

3.2.4.2.4. Passo 3: Orquestração dos serviços com processos de negócio

A orquestração dos serviços com os processos de negócio se traduz em definir quais processos de negócio estão encapsulados em quais serviços e especificar o que pode dar errado na execução e como o sistema irá responder para condições de finalização não esperadas. Esta atividade pode ser realizada através da linguagem BPEL, conforme mostra a Figura 3.9 abaixo:

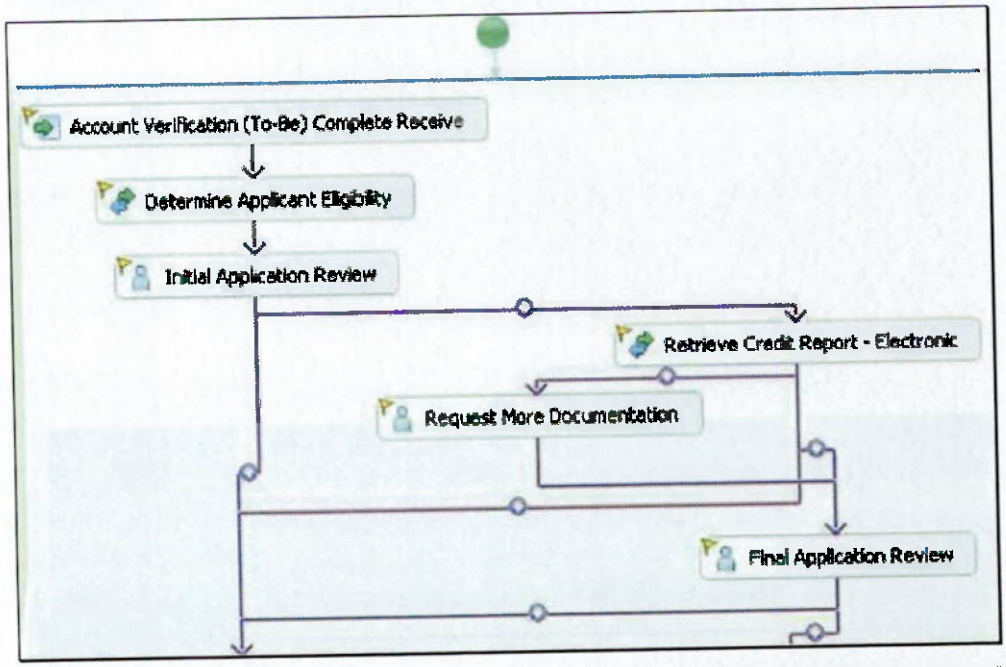


Figura 3.9 - Exemplo de coreografia de serviços utilizando BPEL (BROWN & JOHNSTON ,2005)

3.2.4.3. Desenvolvimento Orientado a Serviço

Esta fase envolve novo desenvolvimento assim como atividades de integração com sistemas pré-existentes, conforme mostra Figura 3.10 abaixo. Suas atividades não são limitadas a software mas também pode envolver subprojetos de infra-estrutura relacionados, tal como projetos de consolidação de hardware ou esforços para centralizar hospedagem de servidores.

Aqui a escolha da plataforma de desenvolvimento – J2EE ou .NET, por exemplo, também se torna efetiva. Esta fase engloba também a fase de testes, sendo que os serviços, por sua natureza genérica de reuso, necessitam ser testados rigorosamente antes de sua implantação em ambiente de produção.

Esta fase do projeto envolve atividades como:

- Desenvolvimento dos serviços
- Integração
- Testes
- Documentação do usuário

Os artefatos principais desta fase são:

- Código base
- Resultados de testes
- Documentação de testes e do usuário

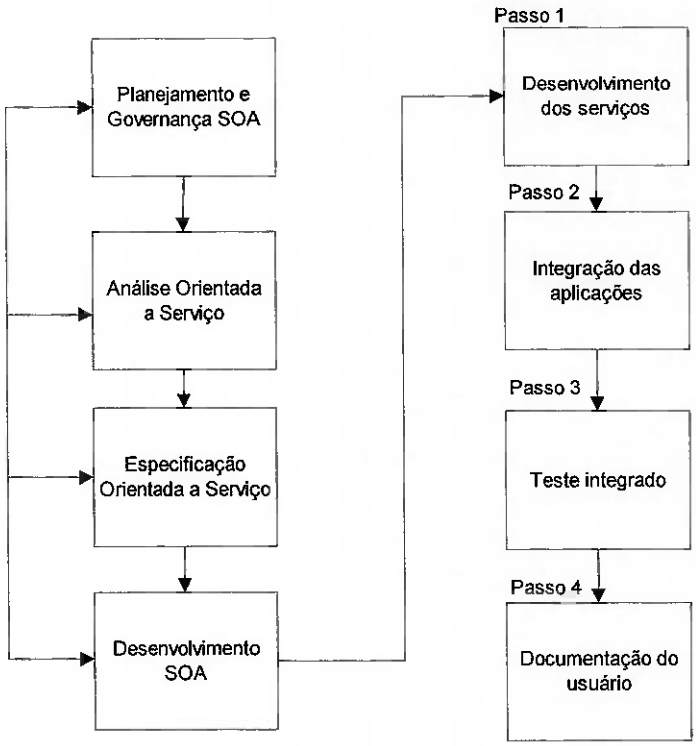


Figura 3.10 – Os passos do processo de Desenvolvimento SOA

3.2.4.3.1. Passo 1: Desenvolvimento dos serviços

O desenvolvedor nesta fase usa alguma ferramenta de desenvolvimento para implementar as especificações e construir os serviços. Algumas ferramentas de fornecedores de tecnologia automatiza a criação de *webservices*, transformando as especificações e coreografias UML, por exemplo, em classes Java, EJBs (*Enterprise Java Beans*) verificando a pré-existência destas classes. As ferramentas também registram os serviços em registros UDDIs e criam a descrição WSDL do serviço procurando e garantindo o reuso de serviços já publicados nos depositórios de serviços.

3.2.4.3.2. Passo 2: Integração das aplicações através de um ESB (*Enterprise Service Bus*)

O papel do desenvolvedor de integração de aplicativos é aplicar a especificação da coreografia dos serviços desenvolvidos como parte dos processos de negócio já transformados e descritos na linguagem BPEL.

BOTTO (2004) define que "ESB, ou Barramento Corporativo de Serviços, é uma camada lógica de mensagens compartilhadas para interoperabilidade entre aplicações e serviços na empresa e entre empresas. Faz transformação inteligente de mensagens e re-roteamento (como *EAI*)". Ainda segundo este mesmo autor, "ESB tem as seguintes características tradicionais e novas:

- Se apoiar em arquitetura orientada a serviços;
- Possui funções de transformação, roteamento e gerência;
- Suporte a diversos padrões de interfaces e conectores;
- Se apoiar em padrões abertos e de segurança;
- Operação e gerência distribuída – e não em um ponto integrador centralizado;
- Ter serviços de repositório;
- Se alavancar em legados de Java e .NET."

NOTT (2004) define que "Um ESB fornece um conjunto de capacidades implementadas pela tecnologia de *middleware*, que permite a integração de serviços dentro de uma arquitetura orientada a serviço", conforme mostra Figura 3.11 abaixo.

Considerando a coreografia dos serviços do negócio como um componente, pode-se estabelecer os seguintes papéis para os demais componentes:

- O **Diretório *ESB Namespace*** fornece informação para permitir o roteamento das interações de serviço.
- O **Diretório *Business Service*** fornece uma taxonomia e detalhes de serviços disponíveis para sistemas participantes de uma arquitetura orientada a serviço.
- **O ESB *Service Gateway*** é usado para fornecer um ponto controlado de acessos externos aos serviços.

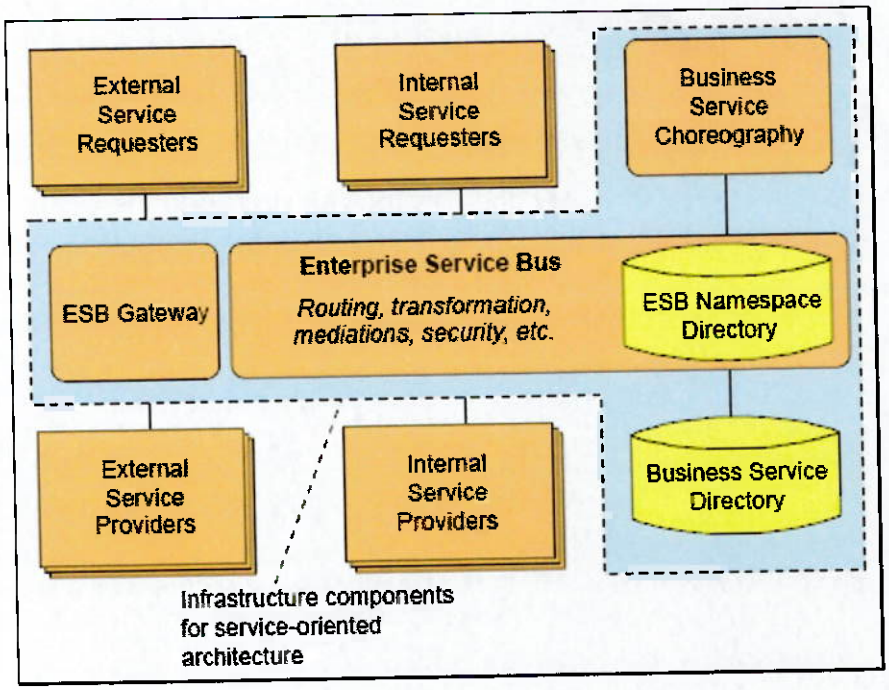


Figura 3.11 - Utilização de um ESB para integração de Aplicativos SOA (NOTT, 2004)

3.2.4.3.3. Passo 3: Teste integrado

Se estamos projetando e implementando uma solução a partir de um nível arquitetural, certamente precisaremos estar aptos a testar desde o topo até a base. Testes unitários resolvem uma parte do problema as não resolvem questões como integração e níveis de sistema, que normalmente são feitos manualmente.

Algumas questões importantes a serem respondidas nesta fase são as seguintes:

- Quais tipos de solicitantes de um serviço podem acessá-lo?
- Quão bem as descrições de serviço comunicam a semântica dos serviços?
- Que tipos de condições de exceção podem um serviço ser potencialmente exposto?

As seguintes dificuldades são encontradas para aplicações distribuídas, mais especificamente aplicações J2EE :

- Testes de aplicações distribuídas são complexos, normalmente requerendo diversas máquinas.
- Difícil simular todas as possíveis condições de falha.
- A utilização de componentes EJB (*Enterprise Java Beans*) J2EE pode dificultar os testes, já que tais componentes são bastante dependentes da infra-estrutura provida pelo servidor.

Para minimizar o impacto destas dificuldades, as seguintes recomendações são feitas na fase de testes :

- Além dos testes dos requisitos funcionais, especial atenção deve ser dada aos testes dos requisitos não-funcionais, principalmente desempenho.
- Em ambientes distribuídos, existem duas possibilidades para a execução de classes de testes :
  - Testes como cliente remoto : é mais simples, mas torna impossível o teste direto de interfaces locais.
  - Testes no próprio servidor : requer maior infra-estrutura, pois normalmente é necessária alguma ferramenta para iniciar os testes. No entanto, permite verificar também as classes com interfaces locais.

3.2.4.3.4. Passo 4: Documentação do usuário

A integração com sistemas sempre é algo complexo. O processo comum exige um profissional com visão macro da necessidade e profissionais de nível técnico de programação mais profundo, fazendo com que a documentação e manutenção destes processos de integração sejam complexas, contudo muito importantes para que eventuais manutenções ou melhorias sejam ágeis e eficazes.

3.2.5. Pós-requisitos do Método SOA-v360

3.2.5.1. Implementação Orientada a Serviço

Nesta fase, você realmente implementa seu SOA.. O artefato chave óbvio para esta fase é sua aplicação em produção. No entanto, mais importante que isso é o plano que estabelece os projetos que pilotarão o SOA na fase pós-implementação:

Algumas questões importantes a serem respondidas nesta fase são as seguintes:

- A infra-estrutura é adequada para atender aos requisitos de processamento de todos os serviços ?
- Como a introdução de novos serviços afetarão serviços e aplicações existentes ?
- Como a introdução de qualquer outro middleware afetam o ambiente existente?

Os principais artefatos chave desta fase são:

- Implantação dos serviços em produção
- Modelo de desenvolvimento continuado
- Modelo de níveis de suporte continuado

3.2.5.2. Gerenciamento, Suporte e otimização pós-implementação

Os sistemas orientados a serviço entraram no ar e tudo corre sem problemas. Acaba a correria e tudo pode ser deixado por si mesmo agora, correto? Errado. Uma vez instalado, seu sistema necessita ser administrado e ser controlado, ambos sob uma perspectiva de TI e uma perspectiva de negócio. A informação reunida durante a fase de gerenciamento é usado para ganhar tempo em processos de negócio, capacitando melhores decisões de negócio e de informação de alimentação no ciclo de vida de desenvolvimento para melhora contínua de processo. Necessitará lidar com edições tal como qualidade de serviço, segurança, e administração geral de sistema. Neste passo, você controla e aperfeiçoa o sistema, achando e

corrigindo imperfeições e problemas. É durante esta fase que você também estará no posto de observação para melhorar não só a arquitetura técnica, mas também a arquitetura de negócio, porque SOA é um processo iterativo. A conclusão desta etapa é o começo de um nova etapa no ciclo. Embora este seja o passo final do ciclo de desenvolvimento de software, é não obstante uma fase muito importante. Esta fase do projeto envolve atividades como:

- Manutenção
- Correção de erros
- Treinamento

3.3. Resumo das Perspectivas, Fases, Atividades e Responsabilidades do Método SOA-v360

Para uma visão mais geral de cada uma das fases do Método SOA-v360, vide os seguintes anexos :

- Anexo A1 – Método SOA-v360: resumo das perspectivas, fases, atividades e responsabilidades do Método SOA-v360
- Anexo A2 – Método SOA-v360: resumo da fase de Planejamento e Governança
- Anexo A3 – Método SOA-v360: resumo da fase Análise Orientada a Serviço
- Anexo A4 – Método SOA-v360: resumo da fase Especificação Orientada a Serviço
- Anexo A5 – Método SOA-v360: resumo da fase Desenvolvimento Orientado a Serviço

4. CONCLUSÕES

A utilização do método integrado SOA-v360, no qual todas as perspectivas – negócio, governança, desenvolvimento e operações - que envolvem uma arquitetura corporativa de TI são levadas em consideração, demonstrou-se bastante adequado para implementar uma arquitetura orientada a serviços (SOA).

Iniciar a modelagem dos serviços sem considerar os requisitos do negócio ou sem a definição dos requisitos não-funcionais da arquitetura pode levar um projeto SOA a não ser bem sucedido. Também importante é que os processos de negócio estejam bem definidos, pois é a partir deles que a composição dos serviços terão início. A especificação da tecnologia e a escolha da plataforma de desenvolvimento é outro fator que deve ser muito bem ponderado, levando-se também em conta as necessidades atuais e futuras do negócio.

Como a implementação de todas as iniciativas mencionadas acima envolvem investimento, na maioria das vezes acontece de nenhuma verba ser liberada pelos executivos sem a comprovação do retorno do investimento (ROI). A utilização dos critérios para cálculo do ROI, como reutilização dos ativos de software e agilidade na adaptação às mudanças dos negócios, foram demonstrados como os principais, mas não impede a utilização de outros que a organização achar mais condizente com sua realidade.

Ainda na perspectiva de governança corporativa, a importância em gerenciar os riscos, levando-se em conta os pré-requisitos como o patrocínio executivo e uma atmosfera de comunicações abertas, tanto no âmbito da equipe de TI quanto dos interessados, leva a organização a se antecipar a eventuais problemas ou falhas inerentes à implementação de uma arquitetura SOA.

4.1. Contribuições do Estudo

Como foi explicado na parte introdutória desta monografia, há muito material que se encontra na *internet* sobre ferramentas e soluções para desenvolvimento e implementação de arquiteturas orientadas-a-serviço SOA, mas nenhuma tem uma visão geral, ou 360° de todos os aspectos que devem envolver um projeto arquitetônico SOA, desde os aspectos de governança e planejamento, até a otimização e melhoria contínua dos processos.

O método SOA-v360 vem preencher esta lacuna, dando um enfoque bem menos mercadológico e focando mais nos aspectos essenciais que levarão ao sucesso na tentativa da implementação de uma arquitetura SOA.

4.2. Limitações e Proposições

O método SOA-v360 aborda, além dos aspectos de governança, os aspectos de análise, especificação e desenvolvimento de serviços. Não há um enfoque maior nos aspectos de implementação e gerenciamento pós-implementação, bem como nos aspectos de gerenciamento da informação e de segurança da informação que podem ser tratados em estudos futuros a respeito deste tema.

4.3. Considerações Finais

Enquanto que a implementação de uma arquitetura orientada a serviços é um investimento de longo prazo a qual necessita de foco sustentado com relação à transformação da forma como a TI é entregue, ela deve corresponder imediatamente às iniciativas do negócio. Conseqüentemente, os benefícios de uma SOA somente serão percebidos se houver um equilíbrio entre os objetivos de longo prazo e necessidades de curto prazo do negócio. Pode-se manter este equilíbrio por meio do estabelecimento de um conjunto de práticas organizacionais, financeiras,

operacionais, de análise, especificação e de entrega desde o início de uma iniciativa SOA.

O modelo consistente que se tem observado é o avanço lento, nos últimos 20 anos, em direção à computação distribuída. O advento da arquitetura orientada a serviços (SOA) não é apenas outra moda passageira, um dentre tantos conjuntos de produtos e serviços: acredita-se que ela terá o mesmo impacto de longo alcance que a Internet teve quando surgiu como empreendimento comercial no início dos anos 90. Olhando para o passado, muitas pessoas não tinham a menor idéia de como esta tecnologia seria usada realmente no mundo comercial.

Assim como a Internet, a SOA terá conseqüências imprevistas para o futuro da computação. Estima-se que a arquitetura orientada a serviços vai forçar uma mudança drástica no equilíbrio de poder no mundo do software, tanto do lado do fornecedor quanto do cliente. O novo modelo pode representar uma mudança radical no modo como o software é criado, reutilizado, recombinado, gerenciado e vendido. Da perspectiva do cliente, a SOA põe o poder nas mãos do usuário de negócio.

SOA cria oportunidades interessantes. Algumas idéias que valem a pena levar em consideração:

- Os clientes vão levar SOA a sério — não só como iniciativa tecnológica, mas também como estratégia de negócio — e demandar que líderes da indústria colaborem para facilitar o trabalho dos clientes.
- Vão surgir novos fornecedores capazes de capitalizar o trabalho realizado por líderes de infra-estrutura em torno de SOA para disponibilizar serviços comercializados que resolvam problemas específicos em mercados verticais específicos.
- Terá início a primeira era da industrialização de software. Os fornecedores vão começar a criar soluções modulares baseadas no modelo SOA para se encaixar em *frameworks* existentes (tais como os volantes que podem ser usados em 20 modelos de carro diferentes).

- SOA abre as portas de um novo mundo, onde "software como serviço" é a norma. Esta transição levará, no mínimo, algo entre 15 e 20 anos. Mas ela é real e terá um impacto surpreendente sobre a indústria inteira, desde modelos financeiros a fornecedores de hospedagem abrangentes.

Muitas empresas estão tentando descobrir se SOA é para valer ou uma moda passageira. Ter-se-á no termo *arquitetura orientada a serviços* um elemento crítico do futuro ou ele será desbancado por novas tecnologias? Se prestarmos atenção à história da indústria de informática, não surpreende que as pessoas fiquem apreensivas. Como se tem observado, tendências vêm e vão rapidamente. Vislumbra-se que esta, porém, é real. Na verdade, testemunha-se o início de uma grande tendência de mercado que terá repercussões no mercado de software na próxima década e adiante.

5. REFERÊNCIAS BIBLIOGRÁFICAS

ARSANJANI, A. *Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA.* IBM developerWorks, Novembro 2004.

ASSAF, N. A. *Finanças Corporativas e Valor.* São Paulo: Atlas, 2003.

BEA. *SOA Resource Center.* Disponível em : <http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/solutions/soa> Acessado em : 10 de dezembro 2006.

BOTTO, R. *Arquitetura Corporativa de Tecnologia da Informação.* Rio de Janeiro: Brasport, 2004.

BROWN, Alan W ; **JOHNSTON**, Simon K. *SOA development using the IBM Rational Software Development Plataform* : A Practical Guide. Setembro, 2005.

ERL, T. *Service-Oriented Architecture: Concepts, Technology, and Design.* Upper Saddle River: Prentice Hall PTR, 2005.

ERL, T. *Serviceorientation.org – About the Principles..*Disponível em: <http://www.serviceorientation.org>. Acessado em: 30 de julho 2006.

GARTNER . *Predicts 2007: SOA Advances.* Disponível em: <http://www.gartner.com>. Acessado em: 30 de dezembro 2006.

GEAO . *Global Enterprise Architecture Organization.* Disponível em: <http://www.geao.org>. Acessado em: 30 de jullho 2006.

IBM. *New to SOA and Web services.* Disponível em : <http://www-128.ibm.com/developerworks/webservices/newto/ >. Acessado em : 19 de novembro 2006.

INGC. *ING Card. An SOA Case Study: SOA Agility in Practice. The Open Group SOA case studies*. Disponível em : <http://www.opengroup.org/projects/soa-case-studies/page.tpl?CALLER=index.tpl&ggid=977>. Acessado em : 10 de dezembro 2006.

KAPLAN, R. S. ; **NORTON**, D. P. *A Estratégia em Ação: Balanced Scorecard.* 6º. Ed., Rio de Janeiro, Campus, 1997.

MICROSOFT. *Alchemy Business Overview*. Disponível em : <http://msdn2.microsoft.com/en-us/architecture/aa973773.aspx>. Acessado em : 10 de dezembro 2006.

NOTT, C. *Patterns : Using Business Service Choreography in Conjunction with an Enterprise Service Bus*. IBM Redbooks Paper. 2004.

NOEL, J. *BPM and SOA: Better Together.* 2005. Disponível em: <http://www-306.ibm.com/software/info/bpmsoa/>. Acessado em: 15 de dezembro de 2006.

OASIS. *Organization for the Advancement of Structured Information Standards Comitê de Especificação. Modelo de Referência para Arquitetura Orientada a Serviço 1.0.* Julho, 2006.

ORACLE. *Strategies for SOA Success*. Disponível em : <http://www.oracle.com/technologies/soa/soa-suite.html>. Acesso em : 19 de novembro 2006.

SCHULTE, W. R. ; **NATIS**, Y. V. N.. *Service-Oriented Architecture*. Gartner, Abril de 1996.

SELVAGE, M., **WOLFSON**, D., **HANDY-BOSMA**, J. *Information management in Service-Oriented Architecture, Part 1: Discover the role of information management in SOA.* IBM developerWorks, Março 2005.

SUN. *Service Oriented Architecture: SOA is not on the horizon, it´s here.* Disponível em : <http://www.sun.com/products/soa/index.jsp>. Acessado em : 10 de dezembro 2006.

ZIFA. *The Zachman Institute for Framework Advancement* . Disponível em : <http://www.zifa.com/framework.html>. Acessado em : 7 de agosto 2006.

6. ANEXOS

Anexo A1 – Método SOA-v360: resumo das Perspectivas, Fases, Atividades e Responsabilidades do Método SOA-v360

Perspectivas	Fases	Quem	Atividades
Negócio	■ requisitos do negócio ■ modelagem do negócio	■ Executivos ■ Analistas de negócio	■ definição dos requisitos do negócio ■ modelagem dos processos de negócio
Governança	■ planejar ■ priorizar ■ gerenciar ■ medir resultados	■ Executivos ■ Gerentes de projeto	■ definição das estratégias e objetivos do negócio ■ definição de indicadores de performance ■ gerenciamento de risco
Desenvolvimento	■ análise ■ especificação ■ desenvolvimento ■ testes	■ arquitetos de software ■ desenvolvedores e integradores ■ testadores	■ análise orientada a serviço ■ especificação e desenvolvimento dos serviços ■ integração dos serviços ■ execução e documentação dos testes
Operações	■ implementação ■ gerenciamento ■ suporte e otimização	■ gerentes de operações do negócio ■ analistas de suporte ■ coordenadores de implementação	■ implementação dos serviços ■ suporte ■ otimização

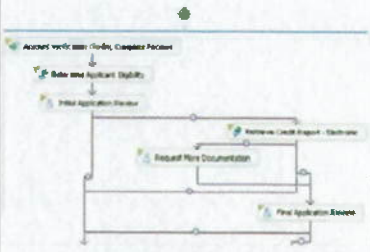
Anexo A2 – Método SOA-v360: resumo da fase de Planejamento e Governança

Passos	Atividades	Artefatos
P1.1. Estabelecer Visão SOA	■ Verificar quais os limites ou fronteiras que a SOA atingirá na organização, bem como quais benefícios (tangíveis e intangíveis) vai se obter com sua implementação	■ Visão SOA
P1.2. Determinar FCS SOA	■ Listar 4 ou 5 Fatores Críticos de Sucesso para o projeto de desenvolvimento SOA	■ FCS SOA
P1.3. Determinar indicadores SOA	■ Desenvolvimento de um mapa estratégico SOA utilizando o BSC(Balanced Scorecard).	■ Mapa estratégico SOA
P1.4. Determinar ROI	■ Através dos indicadores BSC, calcular Retorno do Investimento do projeto SOA	■ ROI do projeto SOA
P1.5. Gerenciamento de Risco SOA	■ Realizar planejamento e avaliação de risco do projeto SOA	■ Análise de risco do projeto SOA

Anexo A3 – Método SOA-v360: resumo da fase Análise Orientada a Serviço

Passos	Atividades	Artefatos
P2.1. Definir escopo da análise e sistemas automatizáveis	■ Identificar quais Unidade de Negócio devem ser consideradas na análise e quais sistemas aplicativos são passíveis de automatização	■ Escopo da análise
P2.2. Identificar e modelar os serviços	■ Decompor (detalhar) os processos de negócio ■ Excluir passos dos processo não apropriados para encapsulação do serviço ■ Identificar os casos de uso para os candidatos à serviço ■ Aplicar os princípios SOA aos serviços candidatos (reusabilidade, p. exemplo)	■ Modelagem dos serviços
P2.3 Definir requisitos não-funcionais	■ Não-Funcionais: requisitos de desempenho e outros atributos de qualidade, assim como requisitos lógicos dos dados e restrições.	■ Requisitos não-funcionais

Anexo A4 – Método SOA-v360: resumo da fase Especificação Orientada a Serviço

Passos	Atividades	Artefatos
P3.1. Especificação da tecnologia e escolha da plataforma de desenvolvimento SOA	■ As principais e mais conhecidas plataformas de desenvolvimento existentes atualmente no mercado são Java da Sun Microsystems e .NET da Microsoft. Elas possuem o foco para o mercado de aplicações corporativas e para a tecnologia *Web Services*.	■ Decisão da plataforma de desenvolvimento 
P3.2.1. Especificação dos serviços – entidades	■ A modelagem dos processos de negócio, geralmente em UML, são utilizadas para especificar os serviços (com operações e atributos)	■ Entidades de serviço 
P3.2.2. Especificação dos serviços – seqüenciamento	■ Seqüenciamento dos serviços, podendo-se utilizar a notação da linguagem UML para o diagrama de sequência.	■ Seqüenciamento dos serviços 
P3.3. Orquestração dos serviços com processos de negócio	■ A orquestração dos serviços com os processos de negócio se traduz em definir quais processos de negócio estão encapsulados em quais serviços e especificar o que pode dar errado na execução e como o sistema irá responder para condições de finalização não esperadas. Esta atividade pode ser realizada através da linguagem *BPEL(Business Process Execution Language)*.	■ Orquestração dos serviços 

Anexo A5 – Método SOA-v360: resumo da fase Desenvolvimento Orientado a Serviço

Passos	Atividades	Artefatos
P4.1. Desenvolvimento dos serviços	■ O desenvolvedor nesta fase usa alguma ferramenta de desenvolvimento para implementar as especificações e construir os serviços. Algumas ferramentas de fornecedores de tecnologia automatiza a criação de webservices, transformando as especificações e coreografias UML, por exemplo, em classes Java, EJBs (Enterprise Java Beans). As ferramentas também registram os serviços em registros UDDIs e criam a descrição WSDL do serviço procurando e garantindo o reuso de serviços já publicados nos depositórios de serviços	■ Serviços desenvolvidos
P4.2. Integração das aplicações através de um ESB (Enterprise Service Bus)	■ Aplicação da especificação da coreografia dos serviços desenvolvidos como parte dos processos de negócio já transformados e descritos na linguagem BPEL.	■ Serviços integrados entre os diversos sistemas / plataformas
P4.3 Teste integrado	■ Além dos testes dos requisitos funcionais, especial atenção deve ser dada aos testes dos requisitos não-funcionais, principalmente desempenho.	■ Documentação dos testes integrados
P4.4 Documentação e manual do usuário	■ Documentação técnica dos serviços, integração dos sistemas e manual do usuário	■ Documentação técnica e funcional

Anexo B – Guia de soluções dos principais fornecedores SOA

Os principais fornecedores de software de negócios do mundo, têm alguma estratégia SOA, alguns mais, outros menos estruturadas. Os fornecedores e suas respectivas soluções SOA que mais se destacam nesta disputa são os seguintes:

Fornecedor	Soluções SOA
BEA Systems	Foram encontradas várias referências e estudos de caso que mencionaram o nome desta fornecedora de software no desenvolvimento e implementação de arquiteturas orientadas-a-serviço. Segundo informações levantadas em BEA (2006), a plataforma *BEA* ***WebLogic Enterprise Platform***™ e a família de produtos ***BEA AquaLogic***™ proporcionam uma infraestrutura de software integrada, desenvolvida em resposta às demandas dos clientes por uma forma mais simples de desenvolver, implantar, integrar e gerenciar aplicações corporativas e Web Services. A inspiração por trás dos produtos BEA tem sido suportar as necessidades dos clientes em transformar seus ativos de TI atuais e legados em uma solução poderosa via web com objetivo de aprimorar os serviços à clientes, parceiros e empregados. A plataforma ***BEA WebLogic*** inclui: - *BEA WebLogic Server*™ - *BEA WebLogic Portal*™ - *BEA WebLogic Integration*™ - *BEA WebLogic Workshop*™ Já a visão da família de produtos BEA AquaLogic é proporcionar uma plataforma completa e otimizada para a implementação de uma arquitetura orientada a serviços (SOA) para toda a corporação. O conjunto de produtos BEA que compõe a família BEA AquaLogic representam um framework necessário para o gerenciamento de serviços (Web Services) em larga escala, independente da tecnologia que o suporta (Java – WebLogic, WebSphere, .NET, SAP, etc.). Os produtos que compõe a família ***BEA AquaLogic*** são: - *BEA AquaLogic Service Bus*™ - *BEA AquaLogic Data Services Platform*™ - *BEA AquaLogic Enterprise Security*™ - *BEA AquaLogic Service Registry*™

Fornecedor	Soluções SOA
IBM	A plataforma com as soluções e softwares para desenvolvimento e implementação da arquitetura SOA é basicamente composta das ferramentas WebSphere/J2EE e Rational e uma das mais completas do mercado. Segundo IBM (2006), a plataforma de desenvolvimento, implementação e governança SOA da IBM utiliza as seguintes soluções e ferramentas: - ***IBM WebSphere Business Integration Modeler***: esta ferramenta permite aos analistas modelar, simular e analisar os processos de negócio. Depois de finalizado o processo, podem transformar estes modelos em UML(*Unified Modeling Language*) e em BPEL(*Business Process Execution Language*) para dar início às atividades de integração. - ***IBM Rational Software Architect***: é uma ferramenta integrada de especificação e construção de aplicações orientadas-a-serviço. - ***IBM WebSphere Integration Developer***: ferramenta utilizada para compor aplicações pelo refinamento dos modelos de processo de negócio . Utilizando o editor que visualiza a BPEL, integradores usam a ferramenta para criar a coreografia dos serviços, podendo testá-las em um ambiente de testes e instalar diretamente em um ambiente de produção. - ***IBM Rational Application Developer***: é uma ambiente integrado de desenvolvimento para automatizar muitas das tarefas construídas no padrão Web services. Desenvolvedores podem focar em escrever a lógica do negócio e automatizar tudo do arquivo WSDL e gerar o código para teste. Com ela, os desenvolvedores podem criar, testar, implementar e publicar as rotinas de Web services. - ***IBM WebSphere Process Server***: pode integrar as diversas infra-estruturas de sua empresa, além de aproveitar os um grande conjunto de aplicações e adaptadores existentes no mercado. - ***IBM WebSphere Message Broker*** : disponibiliza um avançado ESB (Enterprise Service Bus) fornecendo conectividade e transformação de dados para aplicações baseadas em padrões ou não e serviços para sua arquitetura orientada-a-serviço. - ***IBM WebSphere Partner Gateway***: integra processos externos e comunidades parceiras com os processos e infra-estruturas internos, permite um robusto e flexível ambiente B2B, além da segurança das transações com os parceiros. - ***IBM WebSphere Portal***: contém um leque grande de tecnologias de portal que ajudam a desenvolver e manter portais B2C, B2B e B2E.

Fornecedor	Soluções SOA
	- ***IBM Workplace Collaboration Services***: fornece uma comunicação integrada de uso e ferramentas de colaboração capacitando as pessoas a fazer seus trabalhos mais eficientemente – a qualquer hora, em qualquer lugar. - ***IBM WebSphere Application Server***: é o pilar da plataforma e software IBM WebSphere, e um bloco chave de construção da SOA. Com o Java™ 2 Enterprise Edition (J2EE™) e a plataforma de aplicação Web services, disponibiliza alta performance de transações. - ***IBM WebSphere Business Monitor***: disponibiliza uma exposição visual de estados de processos de negócio com alertas e notificações a operadores-chaves que facilitam a melhora contínua de seus processos de negócio. Controlado por um painel customizável como páginas de Portal WebSphere que são visualmente intuitivo, com funcionalidades scorecards e indicadores chave de desempenho. - ***IBM Rational Portfolio Manager***: com esta ferramenta, executivos podem priorizar projetos propostos, existentes e em construção baseados em prioridades, risco e retorno. O produto disponibiliza um ambiente completo de gerenciamento de projeto, colaboração e trabalho. Com um repositório central se torna fácil os gerentes de projeto reportar os status, equipes colaborarem dentro e entre os projetos e decisões serem tomadas mais rapidamente. Adicionalmente, pode-se obter informações financeiras como acessar o retorno financeiro do projeto (ROI) e controlar se os indicadores e metas do negócio estão sendo cumpridas.
Microsoft	Apesar de sua tão conhecida plataforma .Net para desenvolvimento de sistemas baseados em *Web services*, não possui ainda uma estratégia e documentação bem definidas e divulgadas no que se refere à SOA. Segundo informações de MICROSOFT (2006), Alchemy é o nome dado para a sua arquitetura SOA. O projeto *Alchemy* foi iniciado em 2003 e, em 2004 passou a se chamar *Alchemy Common Plataform* (ACP). É uma tentativa de integrar as suas fragmentadas aplicações da linha-de-negócios através da utilização de web services, mas com um enfoque quase total em tecnologia e bem menos em negócios.
Oracle	Baseada em sua "Oracle Fusion Middleware Solutions" da ORACLE(2006) cuja integração também utiliza tecnologia J2EE™, permite às organizações integrarem suas aplicações, apesar do fornecedor, tecnologia e plataforma. As principais soluções compreendem: - ***Oracle BPEL Process Manager***: fornece integração de processos que

Fornecedor	Soluções SOA
	conecta aplicações, sistemas de TI e parceiros de negócio, utilizando a linguagem BPEL para orquestração dos processos de negócio. - ***Oracle Enterprise Service Bus(ESB)***: fornece serviços de mensagens, roteamento e transformação de dados, conectando sistemas através dos adaptadores Oracle. - ***Oracle Business Activity Monitoring(BAM)***: fornece ferramentas para construir interativos e painéis e alertas em tempo-real para monitorar serviços e processos de negócio para a tomada de decisão. - ***Oracle Web Services Manager***: administra a arquitetura-orientada a serviço através de políticas de acessoe validação de conteúdo. Suporta Web services implemntados em qualquer plataforma, incluindo siatemas baseados em Java e .Net. - ***Oracle Business Rules***: cria um nível grande de agilidade para modificar as regras de negócio de acordo com a evolução do negócio, também integrada com fluxos em BPEL.
Sun Microsystems	SUN (2006) é um dos mais representativos fornecedores da plataforma onde a arquitetura SOA pode ser construída, pois é fornecedora proprietária da plataforma de desenvolvimento de sistemas distribuídos Java™ , que concorre diretamente com a plataforma .NET da Microsoft e também é utilizada por outros fornecedores como IBM, Oracle e BEA Systems que construíram suas plataformas proprietária utilizando J2EE™. A plataforma de desenvolvimento e implementação da Sun é baseada no Java Enterprise System (JES). Os componentes que compõe o JES são 8 suítes denominadas de: - ***Sun Java Composite Application Platform Suite (Java CAPS)*** proporciona um leque de soluções de negócio para agilizar os serviços de TI dentro de uma organização, aproveitando os ativos legados existentes sem a necessidade de extensivas codificações. Esta plataforma totalmente integrada e baseada em SOA proporciona um conjunto de integração e composição de aplicações, incluindo BPM e um conjunto vasto de conectores. Esta suite é composta de outras quatro suites: - ***Sun Java B2B Suite:*** permite o gerenciamento eficiente de diversos requisitos "business-to-business" através de cadeias de suprimentos e redes de distribuição. - ***Sun Java ESB Suite :*** avançada ESB distribuída através de padrões abertos de suporte para Web services, XML, transformação e roteamento inteligente que ajuda a simplificar o complexo e custosa infraestruturas de

Fornecedor	Soluções SOA
	mensagens. - ***Sun Java Application Platform Suite:*** endereça os desafios de manuseabilidade, performance, segurança e disponibilidade de aplicações e processos críticos ao negócio. - ***Sun Java Web Infraestructure Suite:*** fornece segurança web, fortalece relacionamentos online entre clientes e parceiros e reduz custos de operações web. - ***Java Identify Management Suite:*** é um ponto focal de uma infraestrtura baseada em SOA, fornecendo um completo fim-a-fim gerenciamento de identificação.