

# Fatores Críticos de Sucesso na Utilização da Arquitetura de Web Services em Micro e Pequenas Empresas Desenvolvedoras de Sistemas de Informação: Uma Pesquisa-Ação

## Resumo

Acompanhar a profusão de novas tecnologias que surgem a todo momento é tarefa árdua para qualquer empresa, principalmente para as de pequeno porte. Em particular, para consultorias e organizações especializadas em desenvolvimento de software, é fundamental que estas estejam atualizadas nas novas tecnologias para que possam se manter competitivas. Este trabalho investigou, através de uma pesquisa-ação, o processo de desenvolvimento de um sistema de informação (DSI) numa pequena *software house*, onde decidiu-se adotar a arquitetura de Web Services (WS) e utilizar uma plataforma que consistia apenas de software livre. Seu objetivo principal era avaliar os fatores críticos de sucesso da adoção de tais tecnologias e padrões no contexto de micro e pequenas empresas desenvolvedoras de software. A pesquisa identificou vantagens e dificuldades não previstas na literatura, sugerindo novas caminhos a serem explorados na difusão das plataformas baseadas em software livre e WS.

## 1. Introdução

A eficácia em obter dados relevantes dentro das organizações se tornou um dos pontos centrais, quase que decisivo, para o sucesso de uma organização. A empresa que consegue disponibilizar informações relevantes, precisas, e em tempo hábil a seus funcionários, clientes e fornecedores, alavanca o seu poder de decisão, minimiza erros e satisfaz seus clientes.

Para atender a demanda cada vez mais crescente em armazenar, recuperar e intercambiar informação, os pesquisadores de Tecnologia da Informação (TI) têm procurado novos métodos e novas técnicas para desenvolver sistemas. A arquitetura Cliente/Servidor e a metodologia de Orientação a Objetos são alguns frutos desta procura. Entretanto as atuais arquiteturas de TI e os sistemas legados das empresas são um obstáculo a estratégias mais ágeis, em função das dificuldades que impõem à introdução de um novo produto ou serviço (Hagel,2004). As arquiteturas de TI atualmente disponíveis dificultam uma transição suave para novas soluções, pois são compostas por sistemas monolíticos e compactos. Constantemente os administradores se deparam com decisões difíceis entre dar continuidade a seus sistemas legados, mantendo a organização “engessada” e sem possibilidade de interagir com novas tecnologias, ou adquirir pacotes integrados, que geralmente envolvem custos elevados e têm um grande impacto nas diversas variáveis organizacionais (ex., estrutura, processos, competências, relacionamentos, clima, etc.). Assim, seguindo o sucesso do paradigma de Orientação a Objeto, as arquiteturas baseadas em componentes e, mais recentemente, baseadas em serviços (SOA – *Service Oriented Architecture*) são algumas das apostas da indústria de software para a viabilização de novas estratégias de desenvolvimento e implantação que visam minimizar a necessidade de mudanças bruscas em seus clientes, e gerar mais rapidamente os benefícios esperados.

*Web Services* (WS) é o resultado de um trabalho realizado pelas maiores empresas de software no mundo, com o objetivo de criar um padrão tecnológico para implementar a arquitetura orientada a serviços. Com a adoção dos padrões abertos e a promessa de investimento contínuo nesta solução feita pelas grandes empresas de software, os departamentos de TI começaram a experimentar e testar a arquitetura de WS a partir de 2002. Segundo uma pesquisa recente realizada entre as maiores empresas do mundo, 37% dos 273 entrevistados disseram já ter sistemas em produção utilizando WS, e 44% estão em processo de pesquisa e desenvolvimento (Westbridge,2004).

Até a presente data, nenhuma pesquisa quantitativa foi encontrada sobre a utilização de WS no Brasil. Contudo, à medida que as empresas multinacionais começam a requisitar soluções que interajam com os WS já implantados, é possível que as empresas brasileiras

também partam para a adoção dessa arquitetura. Assim, as empresas de software e consultorias fornecedoras de serviços de terceirização que buscarem investir nesta tecnologia poderão ter uma vantagem competitiva neste novo cenário.

Esta pesquisa teve como objetivo avaliar novos métodos para desenvolver sistemas de informação (SI) que fossem mais adaptáveis a um ambiente de negócios mais volátil, e que minimizem o custo de desenvolvimento para empresas desenvolvedoras de software. Para tal, utilizou-se o método de pesquisa ação, analisando os fatores críticos de sucesso no desenvolvimento de sistemas de informação baseados em WS de uma pequena software house, e identificando potenciais vantagens e percalços encontrados durante o projeto. Como objetivo secundário, tem-se a avaliação dos benefícios e problemas associados ao uso de ferramentas de desenvolvimento de SI baseadas em software livre como forma de diminuir os custos do processo de construção de software.

O trabalho está estruturado da seguinte forma. Primeiramente, é feita uma breve revisão da história e o avanço das arquiteturas de desenvolvimento de SI, desde a década de 1960. Esta revisão provê ao leitor os conhecimentos básicos para entender o funcionamento da tecnologia de WS em relação às demais arquiteturas e, com isso, avaliar as oportunidades que podem surgir de sua adoção por micro e pequenas empresas. A segunda parte do artigo descreve como a arquitetura de WS foi utilizada por uma microempresa para resolver problemas de negócio e adicionar valor a seus serviços. Por fim, são apresentadas as conclusões obtidas sobre a adequação da arquitetura de WS e os fatores críticos de sucesso de sua adoção em MPes.

## **2. Evolução das Arquiteturas de Sistemas de Informação**

Uma arquitetura de software se preocupa com os aspectos mais gerais dos diferentes tipos de componentes de software utilizados numa organização, e com suas formas de interação (Garlan et al.,1997). Sob o ponto de vista da Administração de Empresas, toda mudança de arquitetura de sistemas de informação pode ser vista como uma tentativa de adaptação dos sistemas de informação de uma organização aos desafios e mudanças do negócio que surgem e não resolvidos de forma satisfatória pela arquitetura vigente. Assim, nas últimas décadas, observou-se o surgimento de várias propostas de arquitetura, geralmente visando atender novas necessidades de negócio. Partindo-se, no final dos anos 60, das arquiteturas centralizadas, baseadas em *mainframes* ou computadores de grande porte, chegou-se, nos anos 80, à arquitetura cliente-servidor, que é geralmente adotada em softwares complexos, como os sistemas ERP (*Enterprise Resource Planing*). Entretanto, o custo para implementar mudanças de rotina ou criar novas funcionalidades nos sistemas que utilizam essa arquitetura ainda é muito grande. Segundo Brown (2004, p.2), a forma rígida pela qual esses programas são implementados, bem como seus códigos difíceis de modificar, geram limitações significativas para a gestão corporativa. As despesas e dificuldades podem ser tão grandes, que algumas empresas preferem abandonar suas iniciativas de alteração dos programas.

Com o advento da *World Wide Web* (WWW ou Web), no início dos anos 90, as organizações encontraram nos protocolos abertos uma nova forma de disponibilizar seus sistemas e serviços. Várias empresas perceberam que a Internet podia se tornar um canal de intercâmbio de informações com seus clientes e fornecedores, incrementando consideravelmente o comércio eletrônico, que, até então, era realizado em plataformas caras e proprietárias, tal como o EDI. Entende-se, aqui, o comércio eletrônico como sendo “a realização de toda a cadeia de valor dos processos de negócio num ambiente eletrônico, por meio da aplicação intensa das tecnologias de comunicação e de informação, atendendo aos objetivos de negócio” (Albertin,1999, p. 15).

Ao mesmo tempo, os navegadores ou *browsers* permitiam que qualquer computador, em qualquer parte do mundo, com qualquer sistema operacional que suportasse os padrões da Web, pudesse acessar aplicativos corporativos desenvolvidos com base nesses mesmos padrões. Expandiu-se o conceito de camadas de software da arquitetura cliente-servidor, definindo-se novas camadas, cada qual responsável por uma função num sistema de informação (ex., montagem das páginas a serem enviadas ao navegador, implementação de algumas das regras de negócio, etc.). Essa nova arquitetura era capaz de reduzir o custo total de manutenção dos sistemas cliente-servidor, além de aumentar significativamente a flexibilidade do processamento e acesso a informações. Com isso, possibilitavam um ajuste mais rápido das empresas às novas necessidades de negócio que surgissem.

Nas últimas décadas, surgiram outras evoluções da arquitetura cliente-servidor. Dentre elas, destacamos a Arquitetura Orientada a Serviços (*Service Oriented Architecture – SOA*) e, mais recentemente, a arquitetura de Web Services, que são descritas abaixo.

## 2.1. Arquitetura Orientada a Serviços (SOA)

SOA (*Service Oriented Architecture*) é a denominação dada a um novo tipo de arquitetura onde softwares e rotinas são disponibilizados como serviços numa rede de computadores (ex., intranets, extranets, Internet), e que podem ser utilizados por diferentes aplicações e para vários propósitos. Idealmente, com este tipo de arquitetura, o desenvolvimento de novas aplicações se resumiria em selecionar os serviços disponíveis e encaixá-los numa determinada seqüência de execução, de acordo com as regras de negócio a serem atendidas.

O desenvolvimento da SOA e da arquitetura baseada em componentes tem suas raízes no conceito de modularidade. Esse conceito é, na realidade, mais geral, indo além da questão específica do desenvolvimento de sistemas de informação, já que a modularidade pode ser entendida com uma estratégia para organizar quaisquer sistemas complexos. Um sistema modular é composto de unidades (ou módulos) que são projetados independentemente, mas que interagem e se comportam como partes de um sistema maior (Baldwin e Clark, 1997). Schilling (2000, apud Iyer et al., 2003) define modularidade como um construto que descreve o grau em que os componentes de um sistema podem ser separados e recombinados. Portanto, a modularidade diz respeito tanto a sistemas cujos componentes estejam fortemente acoplados, quanto ao grau em que as “regras” da arquitetura de um sistema permitem (ou proíbem) a combinação desses componentes (Iyer et al., 2003).

A principal motivação para o desenvolvimento da arquitetura SOA é o problema de manter a estratégia de uma empresa, seus processos de negócio, e a infraestrutura de TI que os suporta sempre alinhados, mesmo quando o ambiente competitivo da organização exige mudanças e correções de rumo freqüentes. A arquitetura SOA permite que novas regras de negócio sejam implementadas rapidamente nos sistemas corporativos, sem que seja necessário realizar intervenções profundas e custosas nos programas e aplicativos. A implementação de novas regras seria realizada simplesmente através da eliminação dos serviços que deixariam de ser utilizados, da inclusão dos serviços exigidos pelas novas regras, ou da reordenação da seqüência de execução no sistema.

## 2.2. Web Services

Desde o final dos anos 80, houve várias tentativas de implementar sistemas baseados em SOA. Entretanto nenhuma das implementações realmente foi à frente, principalmente devido a três fatores: (1) não havia softwares para integração de aplicativos (*middleware*) padronizados e abertos (não-proprietários); (2) não existiam definições de interfaces padronizadas para permitir a conexão entre os módulos; e (3) não havia compatibilidade e integração entre os produtos dos diferentes fornecedores (Stal, 2002).

Na busca por novas formas de utilizar a arquitetura orientada a serviços, foi criado, em setembro de 2000, o grupo de trabalho W3C. Formado por membros das maiores empresas de software do mundo, tais como Microsoft, IBM, Oracle e Sun, o grupo definiu uma nova arquitetura computacional, com condições de melhorar o suporte e aprimorar e agilizar a interação entre processos de negócio, e, por conseguinte, entre empresas (W3C, 2003). Essa arquitetura, denominada Web Services, é baseada no conceito de distribuição e modularidade, adotando protocolos abertos e padronizados para promover a integração de aplicações com baixo acoplamento (*loosely coupled applications*) (Sleeper, 2001; Iyer et al., 2003). Segundo Clabby (2002, p.2), “WS pretende afetar a forma como os sistemas de tecnologia de informação são modelados, distribuídos e comprados”.

Pode-se dizer que a tecnologia mais importante para a implementação de aplicações WS é XML. Formalmente, XML é uma recomendação sobre como os dados trocados por aplicativos devem ser estruturados. Na prática, é uma forma de descrever e compartilhar dados usando um formato comum de apresentação (Clabby, 2002). Por ser definido sob forma de texto, com marcações bem definidas, e em função de sua natureza hierárquica, XML é facilmente entendido tanto por um ser humano, quanto por um programa de computador que precise interpretá-lo.

A padronização definida para WS pode ser dividida em três camadas cujos nomes coincidem com seus papéis conceituais: *Interação*, *Descrição* e *Descoberta* (Kreger, 2003). Para cada uma destas camadas, as regras e funcionalidades de outras três sub-camadas devem ser respeitadas, para que a funcionalidade do modelo completo possa ser alcançada. O modelo e suas camadas pode ser no link <http://www.jsk.com.br/figura1.jpg>.

A primeira camada, *Interação*, refere-se ao processamento físico, onde se dá efetivamente a execução das ações desejadas, e onde se define os protocolos de comunicação padronizados, tais como HTTP e SOAP. Estes protocolos são utilizados para que os dados recebidos e enviados possam ser transportados na forma mais transparente possível pela infraestrutura da Internet e das redes das corporações (intranets e extranets), sem a necessidade de softwares intermediários (*middleware*) ou de qualquer outro artifício técnico. Vale ressaltar que, embora SOAP seja o protocolo mais utilizado na implementação de Web Services, existem outras opções, tais como Xml-RPC.

A segunda camada, *Descrição*, tem como objetivo especificar: (1) as funções que cada serviço pode prestar (descrição da implementação); (2) as informações de entrada necessárias para que o serviço possa ser executado; e (3) os tipos de resultados que devem ser esperados (descrição da interface). A camada de descrição segue uma padronização chamada WSDL (*Web Services Description Language*), que também é definida com base no padrão XML.

Ainda na segunda camada, temos a descrição das funções transacionais e de orquestração ou coordenação. A partir delas, é possível especificar como o serviço que está sendo descrito pode se encaixar num processo de negócio como um todo. A especificação BPEL4WS, por exemplo, que está em fase adiantada de padronização, permitirá que esta camada seja integrada aos pacotes de gerenciamento de processos e às ferramentas CASE.

A terceira camada, *Descoberta*, é descrita pelos padrões UDDI (*Universal Description Discovery and Integration*) e WS-Inspection (*Web Services Inspection Language*). As definições dessa camada permitem que as empresas e agentes envolvidos numa interação possam procurar e descobrir serviços que sejam interessantes para as suas operações. Para isso, já existem organizações se mobilizando para fornecer portais com serviços de busca e diretórios por categoria de serviços, no modelo de páginas amarelas. Um exemplo é o site [www.xmethods.com](http://www.xmethods.com).

Além dessas três camadas conceituais, existem outras três camadas que permeiam e controlam cada uma das camadas acima citadas. A camada de *controle de qualidade* contabiliza as falhas que por ventura ocorrerem, e verifica se o tempo de resposta do serviço está dentro do pré-estabelecido. Já a camada de *segurança* verifica quem está acessando os serviços, e monta os históricos de acesso. Finalmente, a camada de *gerenciamento* permite

que o provedor dos WS possa controlar as estatísticas, históricos e parâmetros do serviço que estão sendo gerados pelas duas camadas anteriores.

Sob o ponto de vista de quem vai apenas acessar os WS, o processo funciona da seguinte forma: a empresa que necessita de um serviço específico busca alguém que presta este serviço, utilizando um software agente ou diretório de WS. Achado o serviço necessário, o agente informa a descrição WSDL para que o software cliente possa obter maiores detalhes sobre o funcionamento do WS. Uma vez de posse da descrição, o cliente contrata o serviço com o provedor do WS (com a possibilidade de que não haja contato humano na contratação), que então lhe dá acesso ao serviço desejado. Isso feito, o cliente pode então obter os serviços do WS. Esse processo está representado na figura disponível no link <http://www.jsk.com.br/ws/figura2.jpg>.

### **3. Modelos e Estratégias de Negócios baseados em Web Services**

Empresários e diretores de TI começam a se voltar para a tecnologia de WS, pois percebem que seus benefícios estão baseados em premissas estratégicas para a empresa. Além de os padrões definidos para a utilização de WS estarem baseados em técnicas que são bem conhecidas<sup>1</sup>, o alto potencial de alcance dos serviços gera oportunidades significativas para o modelo econômico que funciona na indústria de desenvolvimento de software. Isso porque as empresas podem utilizar a sua infra-estrutura atual para implantar uma nova arquitetura de serviços e interligar sistemas heterogêneos (Booch, 2003). Dessa forma, a implementação da arquitetura orientada a serviços utilizando WS traz um grau de flexibilidade que não era alcançado com as tentativas anteriores, baseadas em arquiteturas orientadas a serviços como CORBA e DCOM (Barry, 2003).

Por outro lado, com a arquitetura WS, novos modelos de negócios podem ser criados, ou seja, novos produtos e serviços podem ser fornecidos de forma a promover uma melhor integração com elementos do ambiente externo da organização, e trocar informações que antes estavam isoladas em “silos” dentro das próprias organizações. Sob o ponto de vista das empresas que não têm sua atividade fim na TI, é possível agora empregar a tecnologia de WS para vender os serviços de suas aplicações internas para outras empresas com necessidades de serviços similares. Assim, os departamentos de TI poderão deixar de ser vistos como centros de custos, para tornarem-se centros de lucros (Lim, 2003).

Uma outra das vantagens dos WS é a possibilidade de utilizá-los em conjunto com sistemas legados. Muitas vezes, os sistemas legados são antigas aplicações de software existentes nas organizações que são vitais para o funcionamento do negócio (Sayles, 1996 *apud* Siqueira, 2002). No entanto, o uso de sistemas legado gera dificuldades para a adaptação da infraestrutura de TI a novas demandas do negócio (Brode, 1995 *apud* Kruchten, 2003). Muitas vezes, as regras de negócio embutidas nestes sistemas não são inteligíveis nem para analistas mantenedores do código, devido às inúmeras alterações que o software sofre ao longo de seu ciclo de vida (Gavino, 2000 *apud* Siqueira, 2002). Assim, uma tecnologia que possa adicionar novas funcionalidades a sistemas legados, sem que seja necessário retirá-los de produção ou decifrar milhares de linhas de código de difícil manutenção, pode aumentar a adaptabilidade e agilidade de uma empresa, e, ao mesmo tempo, reduzir significativamente o custo do desenvolvimento e manutenção de sistemas.

Pode-se, portanto, vislumbrar três categorias de empreendimentos que estarão utilizando WS. A primeira delas engloba empresas que irão implantar os serviços em conjunto com seus sistemas legados. Estes serviços estarão confinados às suas redes internas. O aspecto de autenticação e segurança não precisará ser tão rígido, já que os serviços não estarão interagindo com sistemas externos. A segunda categoria está associada a empresas que têm interesse em disponibilizar WS para clientes e fornecedores, e, com isso, melhorar o seu relacionamento com os elos de sua cadeia de suprimentos. Dentro dessa categoria, podemos

citar a Amazon.com, que vem disponibilizando WS para seus fornecedores. Atualmente, já existem mais de 50.000 desenvolvedores utilizando os serviços disponibilizados pela empresa (VUNET, 2004). Na terceira categoria, estão as empresas de software e consultorias que já dispõem de vários produtos em produção e, com isso, têm um potencial imenso a explorar. Estas empresas podem subdividir seus sistemas em pequenos serviços que podem, então, ser vendidos separadamente, utilizando a infra-estrutura da Internet. Desse modo, elas poderão criar novos mercados, compostos por empresas que antes não estavam dispostas a comprar um pacote completo, mas tinham interesse em contratar apenas partes das soluções oferecidas pelas consultorias. Por outro lado, as empresas que irão comprar os WS também obteriam vantagens, pois não seria mais necessário fazer grandes investimentos iniciais para começar a fazer uso dos serviços disponibilizados, podendo-se experimentar diversos tipos de solução a um custo bem mais baixo que o de costume. Principalmente para as micro e pequenas empresas, que não dispõem de muitos recursos para avaliar e testar soluções complexas, adotar uma implementação gradual e pontual pode ser uma ótima estratégia.

#### **4. Metodologia de Pesquisa**

Com o intuito de avaliar as dificuldades, custos, benefícios e fatores críticos de sucesso da implementação de Web Services por micro e pequenas empresas, foi realizada uma pesquisa-ação (Thiollent, 1986, 1997; Checkland, 1991; Baskerville & Wood-Harper, 1996) envolvendo uma microempresa desenvolvedora de software. Um dos autores da presente pesquisa atuou como coordenador no projeto de desenvolvimento estudado, estando diretamente envolvido em atividades de negociação com os clientes, gerenciamento e contratação dos desenvolvedores, e definição de estratégias e ações para o processo de desenvolvimento do sistema de informação.

Ao contrário das análises fundamentadas nas perspectivas positivistas e críticas/materialistas, as análises interpretativas permitem que se desenvolva uma visão contextualizada dos fenômenos sociais e organizacionais (Klein e Myers, 1999). Na medida em que os sistemas de informação envolvem pessoas, regras e normas, funções, interações, etc., além, é claro, de hardware e software, sua introdução nas empresas constituem um fenômeno eminentemente social. Assim, uma das premissas básicas deste estudo é que o conhecimento detalhado sobre tais fenômenos só pode ser obtido através de construções sociais situadas histórica e regionalmente no contexto onde eles ocorrem (Klein e Myers, 1999). Por conseguinte, os métodos de coleta e análise dos dados empregados foram essencialmente interpretativos. A coleta de dados se deu através de entrevistas, observação direta, e de documentos e mensagens de correio eletrônico trocados entre os participantes da pesquisa-ação; a análise dos dados coletados consistiu na busca de temas, categorias e dimensões essenciais que representassem o fenômeno estudado, sob o ponto de vista dos participantes.

Seguindo a metodologia de pesquisa-ação (ex., McKay & Marshall, 1999), o primeiro passo foi recolher e analisar o material bibliográfico disponível, incluindo referências sobre as várias arquiteturas utilizadas no desenvolvimento de SI, os diferenciais competitivos que trazem para uma organização, e experiências de adoção em micro e pequenas empresas, preferencialmente situadas em território brasileiro. A partir daí, buscou-se na literatura, métodos para se avaliar os fatores críticos de sucesso (FCS) em projetos de DSI (Wasmund, 1993; Rothenberger, 1998; Reel, 1999) e indicadores de desempenho de processos de DSI que pudessem ser utilizados para embasar essa avaliação (Colombo, 2002; Chua, 2004; Hörbst et al, 2005). Tais indicadores contemplavam principalmente os aspectos de produtividade do processo de DSI e a qualidade do seu resultado. A partir de então, as seguintes atividades foram realizadas:

1. Tendo por base o referencial teórico e de entrevistas com outros desenvolvedores que já tinham experiência em WS, foram definidos potenciais fatores críticos de sucesso para o uso da arquitetura de WS por MPEs desenvolvedoras de SI.
2. Foi feito um planejamento para a execução do projeto de pesquisa-ação, de acordo com as seguintes etapas:
  1. O projeto de pesquisa foi explicado detalhadamente para os clientes e desenvolvedores e obtido o seu consentimento para a realização da pesquisa-ação.
  2. As ferramentas de desenvolvimento e bibliotecas de classe necessárias para implementar os Web Services foram escolhidas e instaladas.
  3. O ambiente de teste foi montado na empresa desenvolvedora de software; alguns WS que vinham como exemplos nas bibliotecas de classes foram compilados para verificar se o ambiente poderia ser utilizado corretamente.
  4. Foram feitos testes locais e na Internet para verificar a conformidade e qualidade dos WS de exemplo.
  5. Foram escolhidas as funções a serem implementadas no sistema baseado em WS, que poderiam ser utilizadas para validar os potenciais fatores críticos de sucesso identificados anteriormente. Essa escolha foi baseada no equilíbrio entre a utilidade e a complexidade de implementação das funcionalidades requeridas pelos usuários.
3. A partir daí, iniciou-se a análise e programação dos módulos para implementar as funções escolhidas, concomitantemente à avaliação do processo de desenvolvimento.
  1. Foi publicado um site na Internet para divulgar o projeto, convidando outros desenvolvedores a participarem do mesmo. Foram elaborados e disponibilizados por um dos pesquisadores materiais explicativos como forma de facilitar o aprendizado por parte de outros desenvolvedores.
  2. Em paralelo, um dos pesquisadores selecionou e contratou os desenvolvedores para o processo de DSI.
  3. Foram disponibilizados WS para que os outros desenvolvedores pudessem realizar testes.
  4. Os primeiros WS escolhidos foram desenvolvidos<sup>2</sup>.
  5. O modelo dos WS foi disponibilizado para parceiros (outras empresas especialistas em desenvolvimento de software) e suas sugestões e críticas foram documentadas para que se pudesse reavaliar o planejamento e os resultados do processo de desenvolvimento.
  6. Os WS foram concluídos e seus resultados, avaliados. Verificou-se se os indicadores de qualidade e produtividade do processo de DSI se mostraram melhores em relação à arquiteturas anteriores.
4. Avaliou-se se FCS previamente identificados ou outros adicionais ocorreram no projeto.

## **5. Indicadores de Desempenho em DSI**

Para avaliar o impacto da adoção da arquitetura de WS como base para o desenvolvimento de sistemas de informação, incluindo também a plataforma de hardware e software necessárias para tal fim, foram identificados na literatura diversos indicadores de desempenho associados ao a processos de desenvolvimento de software. De forma geral, eles contemplam três diferentes perspectivas: a qualidade do produto desenvolvido, e o custo do processo de DSI, e a produtividade na execução das atividades desse processo. O primeiro grupo contempla os requisitos de qualidade que o contratante deseja que o produto final tenha. Os outros dois grupos estão associados aos requisitos que a empresa contratada precisa atingir para que a sua operação seja lucrativa, isto é, que o custo do DSI seja menor que o preço pago pelo contratante e baixo o suficiente para que o contratante possa pagar.

A questão da qualidade em software é um assunto muito abrangente, com diversas matizes. Na presente pesquisa, adotou-se a visão da ISO/IEC 9126 (Bhatti, 2005), que

também é utilizada pela Comissão de Estudos de Qualidade de Software da ABNT e já foi utilizada por diversos autores (ex. Colombo, 2002; Chua, 2004; Hörbst et al, 2005) para a avaliação de arquiteturas e softwares. A norma define as seguintes categorias básicas para os indicadores da qualidade:

- **Funcionalidade:** indicadores que têm como objetivo verificar se o SI atende às necessidades do cliente.
- **Confiabilidade:** avalia o quanto o sistema está livre de falhas e como funciona a sua tolerância a falhas. Contempla também a recuperabilidade, isto é, o intervalo de tempo necessário para recuperar informações em caso de falha e durante quanto tempo é possível utilizar o sistema continuamente, sem interrupções.
- **Usabilidade:** avalia a facilidade de uso do SI por parte dos usuários.
- **Eficiência:** mede quão eficaz é o sistema, ou seja, a relação entre a quantidade de recursos computacionais necessários e a quantidade de funcionalidades existentes no sistema.
- **Manutenibilidade:** mede a dificuldade em dar manutenção no SI, ou seja, o grau de complexidade e flexibilidade envolvida na alteração e inclusão de novas funcionalidades.
- **Portabilidade:** avalia o nível de adaptabilidade e de independência entre o SI e as tecnologias utilizadas por ele, ou seja, qual o grau de facilidade encontrado ao se tentar utilizar o SI em outras plataformas de hardware e software. Mede também a facilidade de instalação em diversas plataformas.

O segundo grupo de indicadores diz respeito a requisitos de ordem econômica. Eles visam avaliar as prerrogativas da empresa que desenvolve o software, sob o ponto de vista de custo e de produtividade do processo de DSI. Favaro (1996) indica que a empresa deve buscar a qualidade sem perder, entretanto, o foco no cliente e na sua lucratividade. Ao fornecer um serviço de qualidade a um preço que o cliente não esteja disposto a pagar ou oferecer um nível de qualidade além do que o cliente precisa pode trazer graves conseqüências para a saúde financeira da empresa. Assim, é importante que se calcule o custo para desenvolver uma solução utilizando uma dada arquitetura (custos de infra-estrutura de software básico e de hardware, de ferramentas de desenvolvimento, e de mão de obra).

O terceiro ponto que influi no processo de DSI diz respeito à produtividade dos desenvolvedores. Aumentar a produtividade, ou seja, a eficiência da mão de obra, é um dos principais objetivos da empresa que desenvolve software. Um dos principais indicadores que expressam a produtividade na produção do software é a reusabilidade. Segundo Coad (1993), este indicador procura medir quanto do código de um software foi reusável, isto é, qual o grau de utilização de um mesmo modelo ou código fonte para as diversas funcionalidades do software. O autor afirma que quanto maior for a reusabilidade, menores serão os recursos consumidos na produção de um software. A afirmação de Coad é corroborada por diversos autores que também citam que o reuso de código além de aumentar a produtividade também melhora a qualidade do software (ex., ROTHENBERGER, 1998).

## **6. Fatores Críticos de Sucesso em DSI**

A revisão da literatura permitiu que se identificasse diversos fatores críticos de sucesso (FCS) para processos de DSI com base em WS (ex., REEL, 1999; WASMUND, 1993; BOHEM, 1994; BOOCH, 2001; W3C, 2001; STAL, 2002). Eles foram complementados com outros pontos considerados críticos por um grupo de cinco diretores de pequenas e médias *software houses* brasileiras, com vários anos de experiência em projetos de DSI, que foram entrevistados especialmente para esta pesquisa. Chegou-se então à seguinte relação de FCS, que foram posteriormente avaliados na pesquisa:

- **Ferramentas de desenvolvimento adequadas:** Dependendo do orçamento do projeto, determinadas ferramentas não podem ser utilizadas, devido ao seu custo elevado. No entanto, a ferramenta adotada deve ter características mínimas de usabilidade, eficiência e

- funcionalidade.
- Equipe adequada de desenvolvedores: Sem uma equipe adequada e motivada, o processo de DSI não consegue ser realizado.
  - Formalização dos testes unitários: Os testes unitários têm como objetivo aprimorar a qualidade do software gerado e influenciam diretamente os critérios de confiabilidade pois minimizam os erros que seriam encontrados no SI.
  - Divulgação dos WS: A publicação e a conseqüente divulgação dos WS, nos padrões (WSDL e UDDI) e canais apropriados, é parte fundamental da construção de WS, como forma de automatizar e de melhorar a qualidade do processo de DSI.
  - Controle da interoperabilidade: Deve-se ter a certeza que os WS estarão dentro dos padrões e que possam ser acessados por qualquer ferramenta. Este controle visa melhorar principalmente os indicadores de portabilidade e manutenibilidade.
  - Manutenção Incremental: A implementação deve permitir que se atualize com facilidade um WS sem comprometer o funcionamento do resto do sistema. Este fator influencia os critérios de manutenibilidade e de confiabilidade pois minimiza os impactos de mudanças e confiabilidade pois minimiza o tempo em que o sistema tem que ficar indisponível ou “fora do ar”.

## 7. O Processo de Desenvolvimento de WS

O objeto da pesquisa-ação foi um projeto implementado por uma micro empresa do Rio de Janeiro, especializada em desenvolvimento de sistemas de informação. O projeto tinha como objetivo primário habilitar escritórios de contabilidade para que estes tivessem condições de prover relatórios e serviços contábeis através da Internet para seus clientes. O sistema seria desenvolvido inicialmente utilizando uma arquitetura de três camadas baseada na tecnologia Web. Entretanto, para que se pudesse avaliar o potencial da nova arquitetura orientada a serviços, a empresa desenvolvedora decidiu utilizar a tecnologia WS.

O ponto de partida do desenvolvimento do sistema foi um projeto de reengenharia realizado para um escritório de contabilidade específico, que presta serviços para pequenas e médias empresas e para profissionais liberais. O principal processo de negócio desse escritório inclui o recolhimento dos dados de notas fiscais emitidas e de pagamentos realizados por seus clientes (sócios das empresas), e a contabilização destas transações e cálculo dos impostos devidos. Ao final de cada mês, o escritório contábil gera um balanço detalhado das atividades de contabilidade realizadas para cada empresa, e os envia por correio para os respectivos clientes.

No início do projeto, o escritório em questão utilizava um sistema de informação desenvolvido por terceiros para armazenar os dados de clientes e suas movimentações contábeis. Apesar do sistema não ter muita flexibilidade, ele conseguia atender satisfatoriamente as necessidades da empresa, que se resumiam à emissão dos relatórios contábeis para os clientes, e ao cumprimento dos requisitos legais inerentes aos serviços de contabilidade. No entanto, a Diretoria da empresa tinha interesse em expandir sua base de clientes, e achou por bem realizar um estudo dos seus processos de negócio para avaliar os benefícios de um possível redesenho. O estudo identificou três fatores básicos que impactavam o bom andamento dos processos na empresa:

1. O alto custo de manipulação dos dados dos clientes, pois várias etapas dos processos eram executadas manualmente.
2. O envio de informações e a comunicação com os clientes eram precários, pois eram realizados através de documentos em papel, telefonemas, e de forma não sistematizada.
3. Havia uma quantidade razoável de retrabalho, devido ao controle de qualidade ineficiente utilizado pelo escritório contábil para conferir as notas fiscais enviadas pelos clientes. O volume de erros (5% dos itens processados) era considerado alto.

Durante o projeto de reengenharia, verificou-se a necessidade de se criar um sistema de informação (batizado como GOPE – Gerenciamento *On-line* para Pequenas Empresas) que fosse capaz de dar suporte ao fluxo de trabalho do novo processo da empresa. O processo redesenhado está descrito no Diagrama de Atividades modelado em UML que pode ser visualizado em <http://www.jsk.com.br/ws/figura3.jpg><sup>3</sup>.

Diferentemente da maioria dos relatos de implementação encontradas na literatura, onde são utilizados o ambiente Java ou .NET da Microsoft, os módulos do projeto que implementam os WS foram desenvolvidos com o compilador Borland Kylix Open Edition, no sistema operacional Linux, e as bibliotecas FreeCLX, ZeosDB, para acesso a banco de dados, e IndySOAP, para fazer a interface SOAP/XML. Todas as ferramentas e o sistema operacional eram gratuitos, baseados em software livre<sup>4</sup>. A escolha dessa plataforma se deu pelas seguintes razões: (1) a falta de conhecimento de ferramentas baseadas em Java e .NET por parte dos desenvolvedores do projeto; (2) a grande experiência adquirida por eles na linguagem Object Pascal e no ambiente Borland, em projetos anteriores; e (3) a falta de recursos financeiros suficientes para se investir em soluções proprietárias e o treinamento necessário.

A escolha da plataforma de desenvolvimento descrita acima permitiu que os desenvolvedores pudessem utilizar um ambiente e linguagem de programação com que estavam familiarizados para desenvolver os serviços. Por conseguinte, puderam obter um domínio relativo da tecnologia de WS rapidamente. Isso gerou ganhos consideráveis no projeto, em função da redução do seu tempo de conclusão, e por não ter sido necessário investir em treinamento ou na contratação de mão de obra especializada. Tais ganhos são ainda mais importantes no contexto de empresas desenvolvedoras de pequeno porte, que geralmente não dispõem de recursos financeiros suficientes e precisam concluir seus projetos rapidamente, para atender suas necessidades de fluxo de caixa. De fato, se não houvesse a possibilidade de se utilizar ferramentas livres, o projeto não se tornaria viável financeiramente para o cliente e os desenvolvedores. Apenas o custo das soluções proprietárias seria maior do que o custo total do projeto efetivamente proposto e aceito pelos clientes.

O projeto foi iniciado com três pessoas: um analista de sistemas e coordenador do projeto, um programador e uma web-designer. Posteriormente, ainda na fase de análise, outros dois desenvolvedores foram contratados, após uma longa busca por indivíduos que realmente tivessem os conhecimentos necessários para se trabalhar com a plataforma de desenvolvimento selecionada. Nessa etapa inicial do projeto, todos os desenvolvedores estavam muito confiantes e estimulados com a forma como o projeto estava sendo conduzido. Além de sentirem-se motivados pela possibilidade de experimentar com uma nova forma de desenvolver sistemas, eles perceberam que poderia haver uma sincronia entre as classes modeladas e a implementação física dos WS. Na opinião dos analistas, isto poderia tornar o desenvolvimento bem mais fácil e, como disse um deles, “bem mais elegante”.

Uma outra vantagem que pode ser constatada foi a maior liberdade dos projetistas com relação à modelagem a arquitetura do sistema. Utilizando-se WS e os conceitos de modularidade inerentes à arquitetura, pode-se separar claramente a interface gráfica das regras de negócio. Com isso, o projeto e as especificações envolvidas ficaram mais claros e fáceis de serem entendidos. Os desenvolvedores acreditaram que a qualidade do software seria melhor devido a este fator. No entanto, a maior liberdade dos projetistas gerou também acaloradas discussões. Alguns desenvolvedores argumentaram que, se um único serviço fosse alterado, todos os outros teriam que ser testados novamente, e que isso seria muito custoso. Como a questão não ficou resolvida, os desenvolvedores ficaram incumbidos de simular mudanças nas regras de negócio para verificar que modificações teriam que ser realizadas, e se o esforço para realizá-las seria maior que o efetuado em arquiteturas tradicionais.

É importante ainda mencionar que houve dificuldades iniciais de adaptação ao uso do Linux por parte dos desenvolvedores, que estavam habituados ao ambiente MS-Windows. A mera falta de teclas de atalho e de outros pequenos recursos visuais a que os desenvolvedores

estavam acostumados no ambiente MS-Windows (e que são diferentes no ambiente Linux) gerou um claro desconforto por parte dos desenvolvedores. O problema de adaptação teve um impacto inicial razoável da produtividade da equipe, mas foi superado, ainda no primeiro mês do processo de desenvolvimento, pela maior parte dos desenvolvedores.

Houve também algumas dificuldades relacionadas à questões técnicas. Foi necessário contornar uma variedade de incompatibilidades entre o Kylix, a distribuição do Linux escolhida, e o servidor Web Apache, e converter e corrigir erros em bibliotecas livres para que pudessem ser utilizadas com a plataforma de desenvolvimento selecionada. Apesar do conhecimento adquirido no processo, o projeto sofreu um atraso de vários meses em função dessas dificuldades. Um outro problema enfrentado diz respeito à segurança de acesso ao sistema e à forma como foi feito o gerenciamento dos WS. Como as padronizações dos aspectos de segurança em WS (WS-Security) ainda não estavam definidas durante a confecção do SI<sup>5</sup> e não existiam ferramentas para implementar as camadas correspondentes, os desenvolvedores tiveram que desenvolver soluções proprietárias que, no futuro, provavelmente não estarão de acordo com os padrões apresentados pelo consórcio W3C. Assim, espera-se que, em breve, os módulos do software tenham que ser reescritos para que os WS fiquem compatíveis com outros sistemas de informação.

Como em outras arquiteturas tradicionais, os desenvolvedores sentiram falta da conexão entre as ferramentas de modelagem de processos, e as ferramentas de programação. Espera-se que WS não fiquem confinados aos protocolos WSDL, SOAP e UDDI, e que se estendam à integração com ferramentas de modelagem de processos. A IBM já disponibilizou aplicações que utilizam BPEL4WS (*Business Process Execution Language for Web Services* ou Linguagem para Execução de Processos de Negócio para WS). Entretanto, a finalização da de sua linguagem ainda está longe de ser alcançada, pois ainda depende de outras padronizações, tais como WS-Transaction e WS-Coordination (Kreger,2002).

Ao término de um ano e meio, o projeto havia gerado com sucesso os WS necessários ao suporte do novo processo de negócio definido para o escritório de contabilidade. O custo total do projeto até então foi de R\$25.600,00. Ele havia envolvido oito pessoas entre programadores, *web designers*, consultores e fornecedores, e gerado mais de 40.000 linhas de código. Além dos oito Web Services publicados, a equipe deu manutenção em mais de 200.000 linhas de código das bibliotecas auxiliares Indy, IndySOAP, Zeos e WebProvider. Embora ainda fosse necessário desenvolver os programas clientes que acessariam os serviços, o projeto foi considerado bem-sucedido pelas diversas partes envolvidas.

## 8. Análise e Conclusões

A análise dos dados coletados na pesquisa-ação demonstrou que a utilização de Web Services contribuiu para a melhoria dos processos de DSI na empresa estudada, pois pode-se perceber melhorias em grande parte dos indicadores de desempenho definidos anteriormente. Mesmo os indicadores que não apresentaram bons resultados se mostraram deficientes em função das ferramentas de desenvolvimento adotadas, e não em função da tecnologia de WS em si. A Tabela 1 descreve, de forma resumida, como os vários indicadores<sup>6</sup> foram afetados pelos fatores definidos acima (FCS), conforme ocorreram no contexto do projeto estudado. Os sinais positivos (+) indicam a melhora nos indicadores, os negativos (-) indicam a piora e os neutros (o) indicam que não se conseguiu identificar nenhuma alteração aparente ou as vantagens e desvantagens não definiram uma melhora no indicador.

<i>Fatores Críticos de Sucesso</i>	<i>F</i>	<i>Cf</i>	<i>U</i>	<i>E</i>	<i>M</i>	<i>P</i>	<i>C</i>	<i>R</i>
Ferramentas de Desenvolvimento: a opção pelo software livre gerou problemas de compatibilidade e dificuldades de seleção de profissionais capacitados; após as correções, a plataforma de WS trouxe várias dos benefícios esperados	-	+	0	+	-	+	+	+
Equipe Adequada: dificuldade de encontrar profissionais com os conhecimentos necessários; os que efetivamente tinham tais conhecimentos puderam aproveitar as vantagens que a plataforma oferecia; maior liberdade no projeto trouxe problemas na fase de teste		+		+	-		0	0
Formalização dos Testes Unitários: a modularidade e a interface bem definida inerente dos WS, facilitavam a construção de programas de teste que cobrissem grande parte das funcionalidades de cada serviço		0			+			+
Divulgação dos WS: uso das comunidades das tecnologias utilizadas permitiu a solução de problemas no desenvolvimento; ainda há poucos serviços eficazes de divulgação de WS, principalmente no Brasil	+		+				+	
Controle da Interoperabilidade: apesar de problemas de interpretação de padrões, a arquitetura se mostrou mais apta a promover a interoperabilidade do que as soluções proprietárias e arquiteturas de SI anteriores	+					+		
Manutenção Incremental: a arquitetura e plataforma adotada mostraram-se flexíveis e de fácil utilização quando alterações dos WS foram necessárias; detectou-se problemas de compatibilidade com as plataformas utilizadas nas outras empresas quando se tentava instalar novas versões dos WS desenvolvidos, já que a frequência de atualizações dos vários softwares pode ser alta		0	+		+			

*Tabela 1: Impacto nos Indicadores de Desempenho*

Em relação à interoperabilidade, uma das principais objetivos da arquitetura SOA, pode-se perceber claramente que existe uma grande distância entre a literatura e as práticas e ferramentas analisadas. Observou-se que vários dos padrões existentes não estão realmente sedimentados na comunidade de SI, pois têm sido interpretados de diferentes formas pelos desenvolvedores de WS. Esse “relaxamento” dos padrões por parte de alguns fabricantes pode trazer os mesmos problemas já encontrados em arquiteturas anteriores, que acabaram por sucumbir às suas próprias incompatibilidades (cf. STAL, 2002; MILINSKI, 2004).

Conforme a literatura estudada, o uso de WS não se efetiva completamente até que estes serviços possam ser reusáveis e compartilhados entre as pessoas e organização (W3C, 2002). Assim, o pesquisador esperava encontrar outras empresas e parceiros que se interessassem em utilizar os WS desenvolvidos para que se pudesse analisar em que grau o reuso de WS entre empresas poderia afetar ou beneficiar o desenvolvimento do SI. Entretanto, nenhum resultado consistente pôde ser obtido, pois a única empresa que estava disponibilizando seus WS não se interessou em integrar os dois sistemas. Esta constatação está alinhada aos resultados de uma recente pesquisa da EDC (2005), que relatou que mais da metade das empresas consultadas não obtiveram vantagens econômicas na produção de Web Services ou somente fizeram implantações confinadas em uma única unidade de negócio. A pesquisa do EDC afirma ainda: “se um WS estiver confinado em apenas uma aplicação ou em um único departamento, ele tem o mesmo valor que uma antiga aplicação legada tinha. Os aspectos de geração de valor e a redução de custos que podem surgir na utilização de WS advêm exatamente do reuso deste WS em vários departamentos ou clientes” (EDC, 2005).

A utilização de software livre se mostrou viável no caso investigado e contribuiu em vários aspectos para o projeto de DSI, sugerindo que a sua adoção pode ser benéfica para micro e pequenas empresas especializadas em DSI. Entretanto, ficou claro para todos os desenvolvedores envolvidos no projeto e para os pesquisadores que o uso de software livre não é um caminho de mão única. Diferentemente do software proprietário, que não pode ser alterado e já conta com funcionalidades pré-determinadas pelo fabricante, o software livre

está em constante mudança, e esta mudança depende intimamente da interação do usuário com a comunidade que criou o software. Uma dos maiores aprendizados desta pesquisa foi a verificação de que o gerente ou empresário que for utilizar esse tipo de tecnologia deve planejar e reservar recursos para possibilitar esse engajamento, por parte de sua equipe, no desenvolvimento das ferramentas que eles mesmos estarão utilizando. Pode-se concluir, assim, que o software realmente não é grátis, e sim livre, pois existe um custo inerente ao uso para que se possa testar, aprimorar e distribuir os aprimoramentos, sejam eles sob forma de código, de documentação ou de exemplos de como utilizá-lo.

Em contrapartida, o processo de interação com a plataforma de software livre trouxe um entendimento das ferramentas utilizadas no desenvolvimento do SI muito mais profundo que os anteriormente experimentados com ferramentas de código fechado. Ao se ter acesso ao código fonte e contato direto com os autores das bibliotecas utilizadas, os participantes da pesquisa se aprofundaram no conhecimento de XML, WS e da arquitetura SOA o que facilitou o desenvolvimento do SI de forma geral.

Pode-se, portanto, notar que diversas oportunidades para novos modelos de negócios podem brotar com o advento da arquitetura de WS, principalmente para micro e pequenas empresas. Devido ao caráter modular da arquitetura, é possível atrair e conquistar novos clientes, e oferecer novos serviços para os clientes existentes, a partir de investimentos relativamente pequenos, como foi demonstrado no caso estudado.

As organizações que conseguirem desenvolver software com maior agilidade e com maior qualidade poderão ter um diferencial competitivo em relação às que insistirem em manter seu ambiente de TI baseado em metodologias arcaicas e arquiteturas desenvolvidas a mais de 30 anos, tais como a arquitetura Cliente/Servidor. Como a maioria das grandes organizações está caminhando para a adoção de arquiteturas mais modernas, é provável que as empresas que não adentrarem neste novo universo deixem de acompanhar os avanços tecnológicos e acabem não participando de novos projetos que utilizem a tecnologia para acoplar seus sistemas ao novo paradigma orientado a serviços.

## 9. Notas

- <sup>1</sup> Por exemplo, o padrão UDDI pode ser visto como uma evolução do modelo de busca e consulta de APIs, e o WSDL pode ser visto com uma evolução da publicação e especificação de interfaces.
- <sup>2</sup> A metodologia de desenvolvimento adotada foi uma variação do MRDS (Metodologia Rápida de Desenvolvimento de Sistemas) especificada por SILVEIRA (2002). O software JUDE foi adotado como ferramenta de modelagem UML (JUDE, 2004; Furlan, 1998).
- <sup>3</sup> O modelo apresentado contém apenas as funções principais que fecham um ciclo integral do processo (ciclo contábil). No modelo completo, existem atividades secundárias que controlam a atividade do mensageiro que faz a entrega de documentos e cheques entre as partes, o cálculo automático dos impostos com base na atividade fiscal informada pelas notas fiscais, e a conciliação bancária informada pelos sócios. Além disso, existe uma interface que transmite as notas fiscais e as despesas digitadas pelos usuários no GOPE, ao sistema legado do escritório contábil.
- <sup>4</sup> O tipo de licença dos produtos (licença GPL) e o conceito de software livre implicam na liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software desenvolvido (Stallman, 1996). Por conseguinte, a empresa ficou obrigada, ao final do projeto, a disponibilizar os códigos fontes e toda a documentação do projeto, também sob a licença GPL, para o público geral.
- <sup>5</sup> Há diversos padrões definidos pela W3C que ainda não foram completamente implementados. É o caso do WS-Policy e o WS-Inspection, que dizem respeito à coordenação e integração do fluxo dos WS entre as empresas e em relação aos processos de negócios envolvidos. Da mesma forma, o padrão responsável pela segurança (WS Security) ainda não está finalizado. Entretanto, muitas organizações já estão reestruturando sua infraestrutura de Internet, introduzindo a tecnologia de certificados digitais, LDAP e HTTPS, a fim de garantir a integridade, confidencialidade e autenticação dos WS (Datamonitor, 2003). Mesmo assim, o ideal é que a segurança já estivesse embutida dentro dos pacotes SOAP. Além de permitir ganhos de produtividade, isto promoveria a compatibilidade e inter-operacionalidade das aplicações de WS, gerando ganhos maiores a longo prazo.
- <sup>6</sup> Na tabela, os indicadores são representados da seguinte forma: F – Funcionalidade, Cf – Confiabilidade, U – Usabilidade, E – Eficiência, M – Manutenibilidade, P – Portabilidade, C – Custo, e R – Reusabilidade.

## 10.Referências Bibliográficas

- ALBERTIN, Alberto Luis. **Comércio eletrônico. Modelo, aspectos e contribuições de sua aplicação.** São Paulo : Ed.Atlas, 1999.
- ANDREWS, W. **SOA Has Impact on Application Development Outsourcing.** Gartner Group. Disponível em <http://www.gartner.com>. Acesso em 18/04/2004.
- BALDWIN, C.Y. e CLARK, Kim B. **Managing in an age of modularity.** Harvard Business Review. Set/Out 1997.
- BARRY, Douglas. **Web Services and Service-Oriented Architectures.** Elsevier Publishing, New York. 2003. pg.76.
- BASKERVILLE, R.L.; WOOD-HARPER, T.A. **A critical perspective on action research as a method for information systems research.** Journal of Information Technology (1996) 11,235-246.
- BOOCH, Grady. **Web Services: The Economic Argument.** Software Development Magazine. May, 2003. Disponível em <http://www.sdmagazine.com/documents/s=7206/sdm0111d/>. Acesso em 06/2003
- BRODE, M. e STONEBRAKER, M. **Migrating legacy systems.** San Francisco: Morgan Kaufmann Publishing. 1995.
- CHECKLAND, Peter. **Achieving 'Desirable and Feasible' Change: An application of Soft Systems Methodology.** The Journal of the Operational Research Society. Vol.36 No.9. 1985.
- CHEN, Minder. **Factors affecting the adoption and diffusion of XML and Web services standards for E-business systems.** International Journal of Human Computer Studies, Volume 58. Pg.259-279. 2003.
- CHUA, B.B. & Dyson, L.E. **Applying the ISO9126 model to the evaluation of an elearning system.** In R. Atkinson, C. McBeath, D. Jonas-Dwyer & R. Phillips (Eds), Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference (pp. 184-190). Perth, 5-8 Dezembro, 2004.
- CLABBY, J. **Web Services Explained - Solution and Applications for the real world.** New York: Prentice Hall PTR. 2002.
- COLOMBO, Regina. GUERRA A.C. **The evaluation method for software product.** ICSSEA - International Conference Software & Systems Engineering and their Applications. Paris - França. 2002.
- DATAMONITOR. **Real Web Services.** Special report on Web Services. 2003. Disponível em <http://www.datamonitor.com>. Acesso em 30/10/2003.
- EDC; **Web Services Development Survey, Spring 2005.** Disponível em [http://www.evansdata.com/n2/surveys/webservices/2005\\_1/webservices\\_05\\_1\\_xmp3.s.html](http://www.evansdata.com/n2/surveys/webservices/2005_1/webservices_05_1_xmp3.s.html)> Acesso em 21/06/2005.
- FURLAN, José David. **Modelagem de objetos através da UML – the unified modeling language.** São Paulo : Makron Books, 1998.
- GARLAN et al. **Architectural Styles, Design Patterns, and Objects.** IEEE Software, pp.43-52, Jan/Fev 1997
- HAGEL, J.; BROWN J.S. **TI Flexível, a melhor estratégia.** HSM Management 43. Abril, 2004.
- \_\_\_\_\_. **Your Next IT Strategy.** Harvard Business Review, 79. pp.105-113. 2001.
- HORBST, Alexander; Fink, K.; Goebel, G. **The ISO/IEC 9126-1 as a Supporting Means for the System Development Process of a Consumer Health Information Web Service.** HINZ 2005: Fourth Health Informatics Conference; pages: [50-53]. Brunswick East, Vic.: Health Informatics Society of Australia, 2005.
- HUANG, C.D, HU, Q. **Integrating Web Services with competitive strategies: A Balanced Scorecard Approach.** Communications of the Association for Information Systems. Vol.13. 2004.

- IYER, B; FREEDMAN, J; GAYNOR, M. e WYNER G. **Web Services: Enabling Dynamic Business Networks**. Communications of the Association for Information Systems. Volume 11, 2003. pg.525-554.
- JUDE. **A Java/UML Object-Oriented Design Tool**. Disponível em <http://objectclub.esm.co.jp/Jude/jude-e.html>. Acesso em 02/11/2003.
- KLEIN, H.Z.; MYERS M.D. **A set of principles for conducting and Evaluating Interpretive Field Studies in Information Systems**. MIS Quarterly. Vol.23 N.1 p.67-94. Março 1999.
- KREGER, Heather. **Fullfilling the Web Services Promise**. Communications of the ACM. Junho de 2003.
- KRUCHTEN, Philippe. **Using the RUP to evolve a legacy system**. The Rational Edge, Jun/2003. Disponível em <http://www-106.ibm.com/developerworks/rational/library/389.html>. Acesso em 25/03/2004.
- LIM, B., WEN, H. J.. **Web Services: An analysis of the technology, its benefits, and implementation difficulties**. Information Systems Management Journal. Spring, 2003.
- MILINSKI, O.Z. OOLERMAN, M.C. **Second Generation Web Services-Oriented Architecture in Production in the Financial Industry**. OOPLSA 2004, Vancouver, Canada. 2004.
- PORTER, M.E. **Competitive Advantage**. The Free Press, New York NY. 1985.
- POWELL, T.C., DENT-MICALEFF. **Information Technology as Competitive Advantage: The role of Human, Business and Technology Resources**. Strategic Management Journal, Vol.18:5, pg.375-405. 1997.
- REEL, J. S. **Critical success factors in software projects**. IEEE Software v.16 (1999)
- ROTHENBERGER, M. Kulkarni, U. Dooley, K. **Critical Success Factors for Software Reuse Projects**. The 19th Americas Conference on Information Systems. Vol. 19 (1998)
- SAYLES, Jonathan. **COBOL and the business program paradigm**. MicroFocus. Disponível em <http://www.microfocus.com> . Acesso em 30/03/2004.
- SILVEIRA, Denis, SCHMITZ, Eber. **Uma Metodologia de Desenvolvimento de Sistemas de Informações em Empresas de Pequeno e Médio Porte**. ENANPAD, 2002.
- SIQUEIRA, José Roberto B. **Source Inspector – uma ferramenta para extração de regras de negócio em sistemas legados**. Monografia (Mestrado em Informática). UFRJ /IM / NCE. Rio de Janeiro. 2002.
- SLEEPER, B. **Defining Web Services**. San Francisco: The Stencil Group. 2001.
- STAL, Michael. **Web services: beyond component-based computing**. Communications of the ACM. Volume 45, Issue 10, 2003.
- STALLMAN, Richard; **The Free Software Definition**. Disponível em <http://www.gnu.org/philosophy/free-sw.html>. Acesso em 4/12/2003
- THIOLLENT, Michel. **Pesquisa-Ação nas Organizações**. Editora ATLAS. 1a Edição 1997.
- \_\_\_\_\_. **Metodologia da Pesquisa-Ação**. 2ª ed. São Paulo: Cortez Editora, 1986.
- VUNET. **50,000 developers use Amazon web services**. Disponível em <http://www.vnunet.com/News/1154546>. Acesso em 25/04/2004.
- WASMUND, M. **Implementing critical success factors in software reuse**. IBM Systems Journal. Vol. 32 N. 41993.
- WESTBRIDGE Corp. **Web Service Usage Survey**. Disponível em <http://www.westbridge.com>. Acesso em 01/2004.
- World Wide Web Consortium – W3C. **Web Services Activity Statement**. Disponível em <http://www.w3.org/2002/ws/Activity> . Acesso em 05/2003.
- World Wide Web Consortium – W3C. **Web Services Architecture - Working Draft**. Disponível em <http://www.w3.org/TR/2003/WD-ws-arch-20030514> . Acesso em 05/2003.