Informática Pública vol. 7 (1): 47-69, 2005

Sistemas Legados e as Novas Tecnologias: técnicas de integração e estudo de caso

HERBERT LAROCA MENDES PINTO[1]
JOSÉ LUÍS BRAGA[2]

(recebido em 03/09/2004; aprovado em 29/12/2004)

PALAVRAS-CHAVE

Engenharia de software – Sistemas de informação – Sistemas legados – Objetos distribuídos

RESUMO

O objetivo deste artigo é conceituar sistemas legados, ou sistemas herdados, e demonstrar abordagens, métodos e tecnologias atuais para manter estes sistemas críticos em produção, estendendo o seu ciclo de vida, integrando-os a novos sistemas e tecnologias. A importância desses sistemas para as organizações, tanto do ponto de vista estratégico quanto do econômico, justifica o interesse pelo assunto com o conseqüente desenvolvimento de técnicas e tecnologias para a sua integração aos novos sistemas. Um pequeno estudo de caso é também apresentado, demonstrando uma das abordagens para integração.

1. INTRODUÇÃO

Sistemas legados, ou seja, sistemas críticos em uso há determinado período e desenvolvidos com tecnologia supostamente ultrapassada, são peças importantes em uma organização. Na contramão da tecnologia, em constante evolução, estes sistemas costumam entrar em produção já desatualizados tecnologicamente, devido a atrasos no seu ciclo normal de desenvolvimento. Geralmente estes sistemas contêm informações vitais para a organização. Como proceder para manter a funcionalidade dos mesmos, sem provocar um impacto negativo ou causar prejuízos ao funcionamento da área de Tecnologia da Informação – TI, e conseqüentemente causar prejuízos para toda a organização?

A atualização e inclusão de novos sistemas e tecnologias em uma organização é constante, tornando imprescindível que novos sistemas sejam integrados aos antigos, interagindo e compartilhando dados, processos e funcionalidades, no conceito de escalabilidade e modularização. Essa característica está se tornando viável com a utilização de conceitos de orientação a objetos e técnicas que "empacotam" os sistemas e extraem dos mesmos somente as informações e funcionalidades relevantes. Um exemplo oportuno do uso dessa tecnologia deve ser citado: a Internet. Grandes organizações públicas e privadas possuem pon-

[1] herbert.pinto@serpro.gov.br
[2] zeluis@dpi.ufv.br

tos de presença na Internet, valendo-se de um conteúdo multimídia para informação, relacionamento, comércio e integração. Não parece razoável que os sistemas em produção dessas empresas, provavelmente sistemas de gestão integrada críticos sejam profundamente alterados em sua estrutura básica ou substituídos para atender a interface com a Web. O mais provável, e menos custoso, é prover uma estrutura que permita à interface Web integrar-se a esses sistemas, extraindo deles informações relevantes.

É importante salientar que nem sempre a decisão de integrar um sistema legado será simples ou se configurará na melhor opção. As opções possíveis devem ser cuidadosamente analisadas, para evitar insucessos ou perdas financeiras significativas. Este artigo se propõe a discutir a questão, trazendo ao leitor os principais conceitos necessários a uma tomada de decisão aceitável sobre o tema.

2. *Sistemas Legados*

"As empresas gastam muito dinheiro em sistemas de software, e para que elas obtenham um retorno deste investimento o software deve ser utilizado por vários anos. O tempo de duração de sistemas de software é muito variável, e muitos sistemas de grande porte permanecem em uso por mais de dez anos. Algumas organizações ainda dependem de sistemas de software que têm mais de vinte anos de existência. Muitos desses antigos sistemas ainda são fundamentais para as empresas, isto é, as empresas dependem dos serviços fornecidos pelo software, e qualquer falha desses serviços teria um sério efeito em seu dia-a-dia. A esses sistemas antigos foi dado o nome de Sistemas Legados" [Somm03].

Sistemas legados "são aplicações de valor crítico para o negócio em produção nas empresas há cinco ou mais anos" [Umar97], ou "qualquer sistema de informação que resiste significativamente à modificação e à evolução" [Brod95]. Alguns autores consideram ainda, como conceito para sistema legado: "toda aplicação em produção". Analisando esta caracterização, verificamos que a partir da entrada de um determinado sistema em produção, dependendo da tecnologia adotada e do prazo no ciclo de desenvolvimento, possivelmente o mesmo já estará ultrapassado, tornando-se candidato a sistema legado

A visão de um sistema legado como um conjunto formado por software e hardware é adequada para uma análise técnica. No nível sócio-técnico e organizacional esta visão deve ser expandida: um sistema legado contém software, hardware, profissionais, processos, regras de negócio e informações geradas e manipuladas por todo o sistema. Existem profissionais ligados diretamente ao mesmo, como analistas, desenvolvedores e administradores de dados; e existem profissionais que utilizam o sistema, como diretores, gerentes e usuários de um modo geral.

É necessário também focar o lado humano de uma possível mudança tecnológica com a modernização ou mesmo substituição do sistema legado. Esse acompanhamento estratégico se chama "gestão de mudanças" (*change management*). "Gestão de Mudanças" são ferramentas e técnicas utilizadas para atenuar o impacto da mudança tecnológica causado nos indivíduos diretamente ligados à tecnologia [Somm03].

Algumas características podem auxiliar na identificação de sistemas legados: sistemas em produção há mais de 5 anos; hardware e software obsoletos; siste-

mas com mais de 10 mil linhas de código; documentação antiga e desatualizada, não condizente com as funcionalidades e processos atuais do sistema; código-fonte amplamente modificado por diversas equipes ao longo do tempo, com alterações não documentadas; utilização de um sistema de arquivos (seqüenciais ou indexados) ou gerenciador de banco de dados obsoletos; interface com o usuário baseada em caractere; sistema não conhecido em sua totalidade pelos profissionais responsáveis por sua manutenção; os usuários do sistema não são capazes de explicar com detalhes suas funções junto ao sistema e todos os processos que o mesmo executa; regras de negócio inseridas somente no código-fonte do sistema; regras não estão documentadas ou não são conhecidas pela grande maioria da equipe de manutenção.

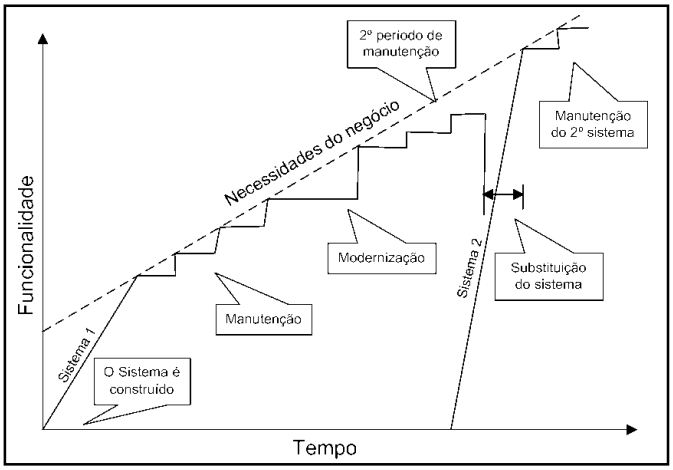

Figura 1 – Ciclo de Vida de um Sistema de Informação. Fonte: [Come00]

2.1 Evolução

A evolução de um sistema é um termo amplo que cobre todo o tempo entre uma simples adição de um campo a um banco de dados até a completa reimplementação de um sistema. Essas atividades de evolução de um sistema podem ser divididas em três categorias, segundo [Weid97]: manutenção, modernização e substituição. A Figura 1 ilustra como as atividades da evolução são aplicadas às diferentes fases do ciclo de vida de um sistema. A linha pontilhada representa o crescimento das necessidades de um negócio, ao passo que a linha sólida representa a funcionalidade fornecida pelo sistema de informação.

A Manutenção é um processo incremental e iterativo em que pequenas modificações são efetuadas no sistema. Essas modificações podem indicar correções de erros ou pequenas melhorias no sistema, e nunca devem indicar grandes mudanças estruturais.

A Substituição é indicada para sistemas que não conseguem se adaptar às necessidades do negócio e para os quais a modernização não é mais possível ou viável.

A Modernização envolve mudanças maiores que a manutenção, mas conserva uma porção significativa do sistema. Essas mudanças freqüentemente incluem a reestruturação do sistema, melhorias funcionais importantes, ou novos atri-

butos de software. A modernização é utilizada quando um sistema legado requer mudanças mais significativas que as possíveis com manutenção. “O tipo de modernização de um sistema legado pode ser definido pelo nível de entendimento do sistema requerido para suportar os esforços de modernização” [Weid97]. Conhecimento interno da lógica do sistema é chamada “modernização white-box”, e a que requer somente o conhecimento das interfaces externas do sistema é chamada “modernização black-box”.

2.1.1 Modernização white-box

A modernização “white-box” requer um processo inicial de engenharia reversa para se entender a operação interna do sistema legado. Componentes do sistema e seus relacionamentos são identificados e uma representação do sistema em alto nível de abstração é produzida [Chik90]. Um entendimento do programa é a principal forma de engenharia reversa usada na modernização “white-box”. Entender o sistema envolve modelar o domínio, extrair informação do código utilizando mecanismos de extração apropriados e criar abstrações que ajudam a entender a estrutura do sistema.

Analisar e entender código antigo são uma tarefa difícil, porque com o tempo todo sistema torna-se ininteligível devido ao provável grande número de manutenções efetuadas. Após o código ser analisado e entendido, a modernização do tipo “white-box” geralmente inclui alguma reestruturação de código ou de sistema. A reestruturação de um software pode ser definida como “a transformação de uma forma de representação para outra, com o mesmo nível relativo de abstração, enquanto se preserva o comportamento externo do objetivo do sistema (funcionalidade e semântica)”. Essa transformação é tipicamente empregada para se conseguir algum atributo de qualidade para o sistema como manutenibilidade ou aumento de performance [Chik90].

2.1.2 Modernização black-box

A modernização “black-box” envolve o exame das entradas e saídas de um sistema legado dentro de um contexto operacional para se entender as interfaces desse sistema. Adquirir o conhecimento de uma interface de determinado sistema não é uma tarefa fácil, mas isso não representa o grau de dificuldade associado à técnica de modernização “black-box”. A modernização “black-box” é freqüentemente baseada em empacotamento. O empacotamento consiste em envolver o sistema legado com uma camada de software que esconde a complexidade não requerida do sistema antigo e recria uma interface moderna. O empacotamento é utilizado para remover as diferenças entre a interface criada por um artefato de software e as interfaces requeridas pelas práticas atuais de integração. Tradicionalmente, empacotamento é uma tarefa de reengenharia, na qual somente a interface do sistema legado é analisada e os códigos internos do sistema legado são ignorados. Infelizmente, essa solução não é sempre a ideal, e freqüentemente se lança mão do entendimento interno do código de um sistema legado, utilizando para isso técnicas “white-box” [Plak99].

As técnicas de modernização “black-box” permitem um aproveitamento e integração dos sistemas legados com menos esforço e recursos computacionais, preservando custos e recursos.

2.2 Arquitetura

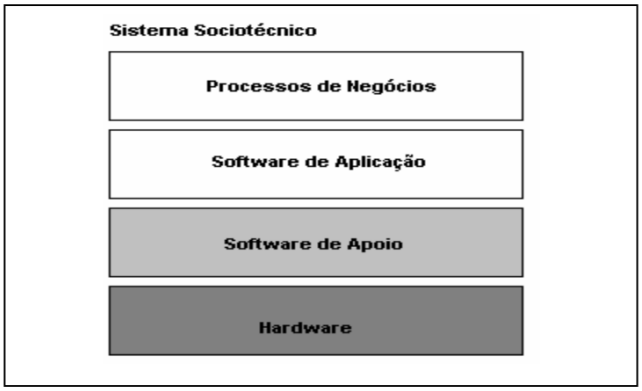

Figura 2 - Modelo em camadas de um sistema legado.
Fonte: [Somm03]

Como visto anteriormente, um sistema legado, no contexto organizacional, possui vários componentes. Para uma discussão técnica, entretanto, podemos considerar os seguintes:

- Hardware de Sistema: na maioria dos sistemas obsoletos o hardware é antigo. Não existem mais fornecedores e a manutenção é dispendiosa.
- Software de Apoio: o software de apoio, como o sistema operacional, compiladores, ferramentas, etc também pode estar desatualizado. O sistema pode ter sido compilado utilizando-se uma versão de um compilador hoje descontinuada, e a mesma deve ser mantida.
- Software de Aplicação: o software de aplicação em um sistema legado não é um único programa de aplicação, mas inclui geralmente vários programas. O sistema pode ter iniciado como um único programa processando um ou dois arquivos de dados mas, ao longo do tempo, podem ter sido implementadas alterações como a adição de novos programas, que compartilham os dados e se comunicam com outros programas no sistema. Os diferentes programas foram escritos por pessoas diferentes que não estão mais disponíveis, e em diferentes linguagens de programação, ou em diferentes versões de uma mesma linguagem de programação.
- Dados de Aplicação: em muitos sistemas legados, um imenso volume de dados se acumulou durante o tempo de existência do sistema. Esses dados podem estar inconsistentes, duplicados em diferentes arquivos. Como acontece com o software de aplicação, os arquivos de dados iniciais sofrem alterações à medida que novas informações são exigidas. Embora ainda existam sistemas legados que utilizam arquivos separados para manter seus dados, um grande número de

sistemas corporativos centralizou seu gerenciamento de dados em um sistema de banco de dados. A vantagem da adoção dessa estrutura é que os dados do sistema são descritos utilizando-se modelos de dados abstratos e é menos provável ocorrer a redundância e duplicidade de dados, facilitando a avaliação do impacto de mudanças no sistema. A separação dos dados de aplicação é importante, como veremos adiante, quando separaremos o sistema legado em camadas, níveis lógicos e verificaremos sua classificação quanto à decomponibilidade. Outra questão a ser resolvida é a obsolescência dos gerenciadores de dados. Há sistemas legados que ainda utilizam gerenciadores de dados hierárquicos ou baseados no modelo CODASYL, que não são mais comercializados e nem são atualizados por seus fabricantes. O custo de migração para um gerenciador relacional, mais moderno, costuma ser muito alto, o que pode inviabilizar uma modernização.

· Regras ou processos de negócios: são as informações sobre os processos internos da organização, codificadas em uma linguagem de programação e espalhadas pelos programas que fazem parte do sistema. Na grande maioria dos sistemas essas regras não estão satisfatoriamente documentadas, sendo do conhecimento tácito de gerentes, analistas e usuários do sistema. Um novo sistema dificilmente conseguirá reproduzir todas essas regras e funcionalidades.

Estendendo o conceito de camadas, podemos classificar um sistema legado quanto à sua separação em níveis lógicos para posterior tratamento, manutenção ou integração, de acordo com a Figura 3:

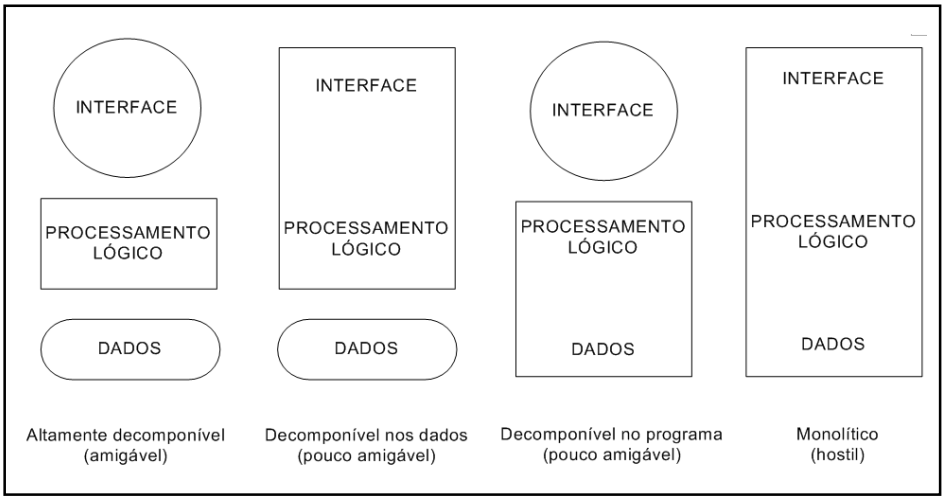

Figura 3 – Níveis lógicos de sistemas computacionais. Fonte: [Mece99]

- Altamente Decomponível (amigável): o sistema é dividido em três níveis lógicos bem distintos e estruturados. A alteração de um nível não afetará significativamente o outro nível. Todos os níveis são prontamente acessíveis. Provavelmente foi utilizada a tecnologia de Sistemas Gerenciadores de Banco de Dados. Sua integração a outros sistemas é considerada fácil.
- Decomponível nos dados (pouco amigável): o sistema é dividido em dois níveis lógicos: o nível de interface em conjunto com o nível de processamento lógico e o nível de dados. O acesso aos dados nesse tipo de sistema é imediato, ao contrário do acesso à interface ou o processamento lógico. Sua integração a outros sistemas é conseguida com algum esforço de programação e utilização de tecnologias.
- Decomponível no programa (pouco amigável): o sistema também é dividido em dois níveis lógicos: o nível de interface, separado e o nível de programas e dados, em conjunto. Os dados somente são acessíveis a partir de funções do sistema (provavelmente transações). Nesta categoria se encontram a maioria dos sistemas legados. Sua integração a outros sistemas é conseguida com algum esforço de programação e utilização de tecnologias.
- Monolítico (hostil): o sistema é composto de um único bloco lógico. São aplicações bem antigas, e o acesso às suas informações é bastante difícil. Sua integração a outros sistemas é bastante complexa. [Mece99]

2.3 Avaliação

Quando se está avaliando um sistema legado, deve-se considerá-lo sob duas perspectivas diferentes, segundo [Warr98]. A partir de uma perspectiva de negócios, deve-se fazer uma avaliação do valor desse sistema para a empresa. A partir de uma perspectiva de sistema, deve-se fazer uma avaliação da qualidade do software de aplicação e do software e hardware de apoio do sistema. A combinação do valor de negócios com a qualidade do sistema é, então, utilizada para ajudar a informar a decisão sobre o que fazer com o sistema legado.

Essas avaliações são meramente subjetivas e sua quantificação deve ser efetuada utilizando-se uma abordagem de questionamentos aos vários indivíduos envolvidos no processo, colhendo assim vários pontos de vista. Dados quantitativos coletados também poderão auxiliar na avaliação do sistema, como: o número de pedidos de modificações no sistema; o volume de dados utilizados pelo sistema, etc. A formulação das questões a serem efetuadas deve ser inerente às duas perspectivas adotadas para a avaliação do sistema legado. Para cada questão deverá ser dado um valor, dentro de determinada faixa, que indicará a situação de determinada característica. Por exemplo, em uma perspectiva de sistema, o questionamento: “O desempenho do sistema é satisfatório?” poderia receber valores de 1 a 5, indicando: péssimo, ruim, satisfatório, bom e ótimo. Estes valores seriam computados (a média, por exemplo) e comporiam um indicador que expressaria a qualidade do sistema.

Para ilustrar essa avaliação, vamos supor que uma organização possua dez sistemas legados. A qualidade e o valor de negócios de cada um desses sistemas são avaliados e comparados com outros sistemas, mediante um gráfico, que mostra o valor de negócios relativo e a qualidade do sistema, Figura 4. Podemos verificar, pelo gráfico, que existem quatro grupos de sistemas:

- Baixa qualidade, baixo valor de negócios. Manter esses sistemas em operação será dispendioso e a taxa de retorno de investimento para os negócios será bastante pequena. Esses sistemas são candidatos a serem descartados.
- Baixa qualidade, alto valor de negócios. Esses sistemas estão prestando uma importante contribuição à empresa e, assim, não podem ser descartados. Contudo, sua baixa qualidade significa que os custos operacionais são altos, de modo que são candidatos à transformação ou à substituição do sistema, se um sistema adequado estiver disponível.
- Alta qualidade, baixo valor de negócios. São sistemas que não contribuem muito para os negócios, mas cuja manutenção pode não ser muito dispendiosa. Não vale o risco substituir esses sistemas, de modo que a manutenção normal pode ser continuada ou eles podem ser descartados.
- Alta qualidade, alto valor de negócios. Esses sistemas devem ser mantidos em operação, mas sua alta qualidade significa que não é necessário investir na sua transformação ou substituição. Deve-se continuar com a manutenção normal do sistema.

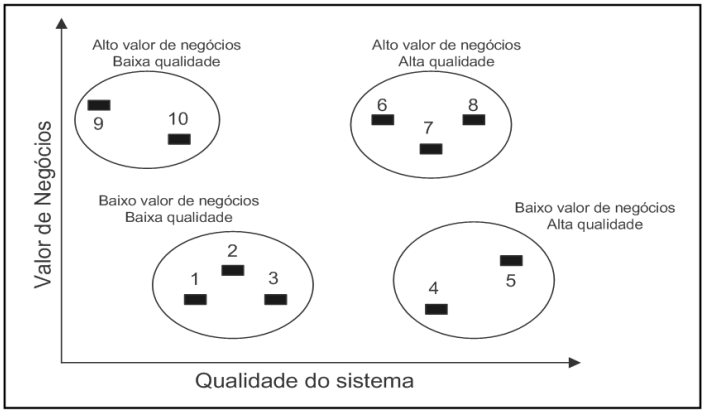

Figura 4 - Qualidade de Sistema e valor de negócios. Fonte: [Somm03]

O ideal é que seja utilizada a avaliação objetiva para informar as decisões sobre o que fazer com um sistema legado. Contudo, em muitos casos, essas decisões não são realmente objetivas, mas se baseiam em considerações organizacionais ou políticas. Por exemplo, se duas empresas fazem uma fusão,

os sistemas utilizados pela empresa com maior influência serão utilizados naturalmente pela empresa resultante e os outros sistemas poderão ser descartados. Se a gerência sênior em uma organização tomar a decisão de migrar para uma nova plataforma de hardware, então isso poderá exigir que aplicações sejam substituídas. Se não houver nenhum orçamento disponível para a transformação do sistema em um determinado ano, então a manutenção do sistema poderá continuar, embora isso venha a resultar em custos de longo prazo mais elevados.

É imprescindível para uma organização efetuar uma correta avaliação de seus sistemas legados. Caso essa avaliação seja efetuada sem critério, as decisões tomadas poderão representar perdas significativas de dinheiro e recursos.

3. ABORDAGENS PARA A INTEGRAÇÃO

Serão apresentadas três abordagens principais para a integração e modernização de sistemas legados. A fragmentação ou empacotamento de telas (*screen scrapping*), a integração ponto a ponto e o empacotamento utilizando-se empacotadores orientados a objeto.

3.1 Fragmentação de telas – Screen Scrapping

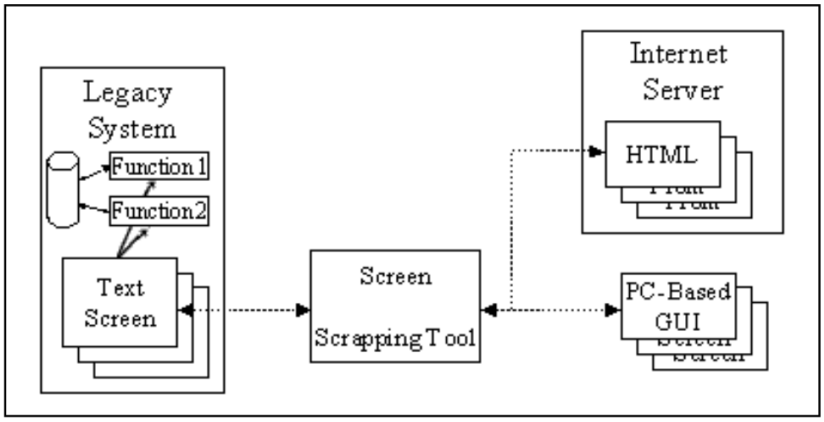

Figura 5 - A técnica "Screen Scrapping ". Fonte [Come00].

O objetivo desta técnica é "empacotar" a interface antiga do sistema legado, geralmente baseada em telas-texto no formato caractere visualizadas via terminal, e fornecer uma interface mais recente e tecnologicamente integrável. Esta nova interface será mais flexível que a antiga, permitindo seu uso em computadores pessoais, criando uma nova interface gráfica ou fornecendo conteúdo para a Web, por exemplo.

Com a fragmentação de telas um programa (*screen scraper*) se conecta a uma aplicação legada e simula os sinais de teclado que um usuário faria para acessar a funcionalidade da aplicação legada. Do ponto de vista de uma aplica-

ção legada, é como se um usuário estivesse utilizando a aplicação. O programa *screen-scraper* utiliza API'S fornecidas pelo software de emulação de conexão do terminal para manipular certas informações de telas predefinidas de uma forma invisível para o usuário. Na maioria das vezes, somente um pequeno subconjunto de instruções da API é utilizado, como *connect*, *wait for string*, *send key*, apenas para nomear alguns exemplos. Na maioria dos casos, o estado da sessão do terminal não é informado ao usuário.

O maior problema em utilizar a abordagem de fragmentação de telas é causado pela ineficiência para manipular eventos inesperados relacionados ao ambiente de conexão com o servidor, como teclado travado, desconexões de sessão, e mensagens de *broadcast* dos servidores [Come00].

3.2 Ponto a Ponto

Uma segunda alternativa para o empacotamento das aplicações legadas é tomar uma abordagem ponto a ponto na qual o empacotador reside tanto na aplicação legada quanto na aplicação não-legada que se deseja integrar. Os dois componentes se comunicam através de chamadas de transação. A idéia básica, como mostrado na Figura 6, é em primeiro lugar escrever o código do empacotador que acessa diretamente a funcionalidade proporcionada pela aplicação legada, chamando as suas sub-rotinas internas. Após isso, deve-se escrever o código do empacotador na aplicação não-legada, que se comunica com o código do empacotador legado enviando suas transações. É muito comum ver o fluxo de transações fluindo em ambas as direções, normalmente porque um componente da aplicação legada foi substituído, completa ou parcialmente, pela aplicação não-legada. A maior vantagem do empacotamento ponto a ponto é a rapidez, pois o empacotador legado acessa o código atual que realiza a funcionalidade do negócio. A maior desvantagem é que o código do empacotador existe para ambas as aplicações, a legada e a não legada, acoplando efetivamente o código do empacotador ao núcleo da aplicação legada. Isso também implica ter desenvolvedores para ambos os ambientes [Ambl98].

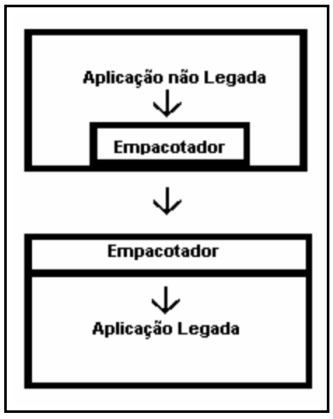

Figura 6 – Empacotando uma aplicação legada através de uma abordagem ponto a ponto. Fonte: [Ambl98]

3.3 Empacotamento Orientado a Objetos – Wrapping OO

A tecnologia de objetos vem sendo utilizada para implementar com sucesso sistemas de computação complexos. Sistemas orientados a objeto podem ser desenvolvidos e implementados de uma maneira mais produtiva, e adicionalmente o uso de recursos como abstração, empacotamento, herança e outras técnicas de orientação a objetos tornam um sistema orientado a objeto simples de ser entendido. Empacotar utilizando tecnologias orientadas a objeto é uma técnica na qual as interfaces de uma aplicação não orientada a objetos são envolvidas ou encapsuladas por uma camada ou casca que possibilita um meio padrão e formal de acesso à aplicação legada por outras aplicações tecnologicamente mais atualizadas e orientadas a objeto. É através dessa nova camada orientada a objeto que o sistema legado irá integrar-se e interagir com novas aplicações, para desempenhar as funções necessárias ao funcionamento do mesmo. Após encapsulada, a funcionalidade do sistema legado torna-se reutilizável nos vários outros novos sistemas que venham a ser integrados ao antigo sistema, como se fosse um objeto comum. Essa é a maior vantagem sobre os outros tipos de abordagens. Do ponto de vista de um objeto do sistema, o empacotador se parece com qualquer outro objeto deste contexto, como mostra a Figura 7.

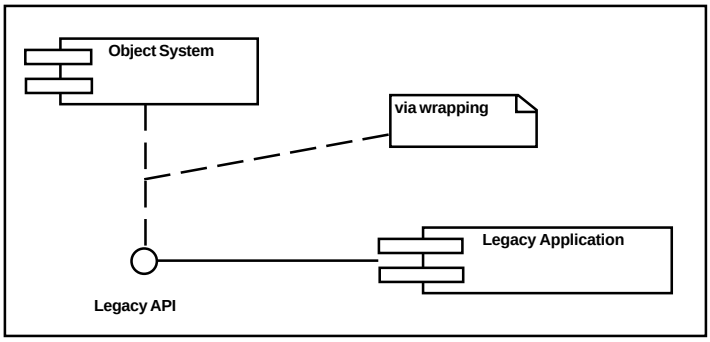

Figura 7 – O processo de empacotamento. Fonte: [Asma00]

4. TECNOLOGIAS PARA A INTEGRAÇÃO

4.1 Corba

A arquitetura CORBA, que significa Common Object Request Broker Architecture, é uma abordagem para objetos distribuídos recomendada pelo Grupo de Gerenciamento de Objetos (Object Management Group – OMG), um consórcio de organizações que estão trabalhando juntas para desenvolver um conjunto de padrões para a computação de objetos distribuídos. A arquitetura CORBA basicamente define os serviços realizados por um ORB (Object Request Broker), que é uma tecnologia de *middleware* que permite que objetos enviem mensagens para outros objetos através de uma rede, proporcionando uma definição de linguagem de interface (IDL) que especifica a abordagem-padrão para a definição da interface de objetos. A IDL CORBA proporciona um mecanismo de empacotamento de objetos que esconde definitivamente os detalhes de implementação de um objeto. A arquitetura CORBA define os padrões que as

empresas que desenvolvem os ORBs necessitam para que os seus objetos possam interagir uns com os outros de maneira consistente. Para um entendimento da estrutura da arquitetura CORBA é necessário conhecer e entender os seguintes componentes:

- Object Request Broker, que permite aos objetos uma transparência no envio e recebimento de solicitações e respostas em um ambiente distribuído. Ele é fundamental para a construção de aplicações de objetos distribuídos e para interoperabilidades entre aplicações em um ambiente hetero e homogêneo.
- Serviços de Objeto, uma coleção de serviços (interfaces e objetos) que fornece funções básicas para uso e implementação de objetos. Os serviços são necessários para a construção de qualquer aplicação distribuída e são sempre independentes dos domínios das aplicações. Como exemplo, o serviço de ciclo de vida define convenções para criar, apagar, copiar e mover objetos. Ele não dita como os objetos são implementados em uma aplicação.
- Serviços Comuns, um conjunto de serviços que as aplicações poderão compartilhar, mas os quais não são fundamentais como os serviços de objetos. Por exemplo, um gerenciamento do sistema ou um serviço de *e-mail* podem ser classificados como serviços comuns.
- Objetos de Aplicação, que são os produtos que rodam em cima da arquitetura CORBA. Objetos de aplicação correspondem à noção tradicional de aplicações, e eles não são padronizados pelo OMG. No nosso contexto um objeto de aplicação é o sistema legado.

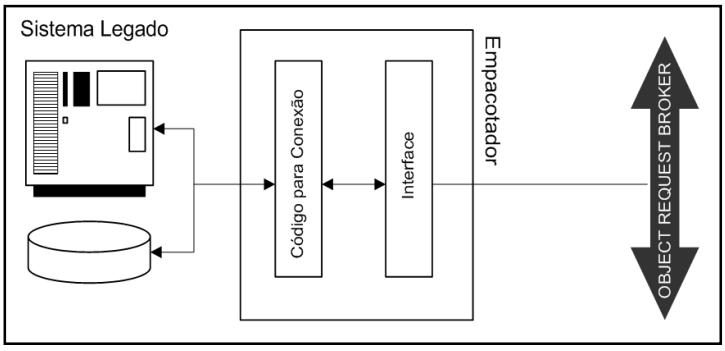

Figura 8 – Um objeto empacotador. Fonte: [Juric00]

4.2 EJB (Enterprise JavaBeans)

Enterprise JavaBeans são componentes fornecidos em conjunto com um servidor de aplicação, ou servidor EJB, que implementa o ambiente de execução para o componente e gerencia serviços comuns, como segurança, transações, estados, compartilhamento de recursos, persistência automática e chamada remota. Isso permite ao desenvolvedor dos componentes o foco somente no problema do negócio a ser resolvido. O primeiro passo para empacotar um sistema

legado utilizando EJB é separar a interface do sistema legado em módulos consistindo em unidades lógicas. O grau de dificuldade em dividir o sistema legado em funções discretas irá variar dependendo do grau em que essas separações foram definidas nas interfaces do sistema legado e quais interfaces novas devem ser construídas. Embora uma abordagem "black-box" seja mais indicada, uma documentação e interface pobre podem tornar necessária uma verificação mais detalhada do sistema legado para uma melhor compreensão do mesmo. O próximo passo é construir um único ponto de contato para o sistema legado. É uma boa idéia centralizar toda a comunicação em um único software. O método de comunicação usado por este software depende de cada situação. Opções para a comunicação entre o ponto de contato e o sistema legado incluem RMI – Remote Method Invocation (protocolo exclusivo para comunicação entre objetos Java) sobre IIOP – Internet Inter-Orb Protocol (protocolo que permite a execução do padrão CORBA sobre o protocolo Web), *sockets* (software para conexão ao protocolo de rede), ou mesmo um *middleware* (um programa capaz de permitir a comunicação entre dois sistemas heterogêneos) orientado a mensagem (MOM – Message Oriented Middleware), que possui a vantagem de separar o servidor EJB do sistema legado e permitir comunicação assíncrona. Este ponto único de contato pode ser implementado como um *bean* (componente), chamado adaptador, ou um *service broker*, um componente de software externo ao servidor EJB. Colocar o ponto de contato dentro ou fora do servidor depende principalmente do método de comunicação escolhido e de algumas restrições de segurança. Por exemplo, se for necessário criar uma nova *thread*, que são blocos de códigos do mesmo programa executados independe e concorrentemente, ou verificar um *socket*, o ponto de contato deve estar fora do servidor de aplicação porque a especificação EJB não permite o multithread de JavaBeans (execução simultânea de pedaços de código JavaBeans) ou a "escuta" de *sockets* [Come00].

A etapa final no empacotamento do sistema legado é implementar um componente empacotador para cada módulo do sistema legado. Na Figura 9 esse empacotador é mostrado como Bean 2. Esses componentes fazem requisições ao sistema legado utilizando um ponto de contato único, de um modo similar ao empacotamento de objetos. Existem várias vantagens na abordagem de empacotamento de sistemas legados utilizando-se componentes. Primeiro, com um esforço relativamente limitado, as vantagens dos sistemas baseados em componentes são fornecidas. Por exemplo, pode-se construir novos JavaBeans que usem os empacotadores de modos não-previstos, aumentando a flexibilidade do sistema. Segundo, os empacotadores são Enterprise JavaBeans e podem ser totalmente integrados com todos os serviços e facilidades gerenciais incluídos no servidor de aplicação. Finalmente, o empacotamento das lógicas de negócio legadas permitem a criação de um roteiro para a substituição do sistema legado incrementalmente. Após empacotar a funcionalidade do sistema legado, pode-se reimplementar os empacotadores um por vez (Bean 1 na Figura 9) , sem necessitar uma substituição total do sistema. Isso é possível devido ao sistema e os clientes não experimentarem nenhuma interrupção, enquanto os empacotadores reimplementados mantêm as mesmas interfaces fornecidas pelo empacotador anterior. Desse modo, é possível a substituição completa do sistema antigo [Come00].

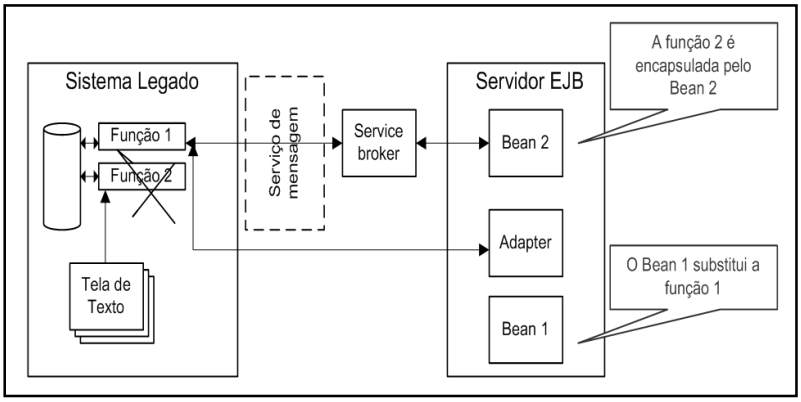

Figura 9 – Empacotando a lógica de negócios legada utilizando EJB.
Fonte: [Come00]

4.3 Web Services

Uma definição de Web Services, segundo o W3C (World Wide Web Consortium): "Um Web Service é um sistema de software desenvolvido para fornecer uma interação máquina-a-máquina sobre uma rede. Ele possui uma interface descrita em um formato padrão (especificamente WSDL). Outros sistemas interagem com o Web Service do modo descrito em sua interface usando mensagens padrão SOAP, tipicamente com http, e uma serialização XML em conjunto com outros padrões Web" [W3C03].

Web Services são interfaces independentes de plataforma que permitem a comunicação entre aplicações utilizando as tecnologias existentes na Internet, como HTTP e XML. As aplicações já estavam aptas a comunicar-se pela Internet há anos, mas somente recentemente foram criadas normas para permitir uma comunicação padrão entre as aplicações distribuídas. Os Web Services dependem de três padrões-chave para permitir essa comunicação, independente da plataforma na qual estão sendo executadas as aplicações, ou linguagem de programação utilizada para o seu desenvolvimento:

- SOAP – Simple Object Access Protocol. Especifica um formato para as mensagens passadas entre os Web Services.
- WSDL – Web Service Description Language. Descreve o web service, permitindo que outros Web Services saibam como acessá-lo, o que mandar como entrada e o que esperar como saída.
- UDDI – Universal Description, Discovery and Integration standard. É um sistema de registro que permite que os Web Services publiquem documentos WSDL, para que outros Web Services. UDDI também fornece a especificação sobre os formatos de entrada de dados, modelos de segurança, protocolos e formatos de saída de dados.

SOAPConstruídos com os fundamentos fornecidos por estes padrões estão surgindo outros padrões para manipulação de processos de negócio complexos, autenticação e integridade de mensagens, etc. Por trás de todos estes padrões está o uso do XML como formato de mensagens e um protocolo padrão como o http para método de transporte [Nave03].

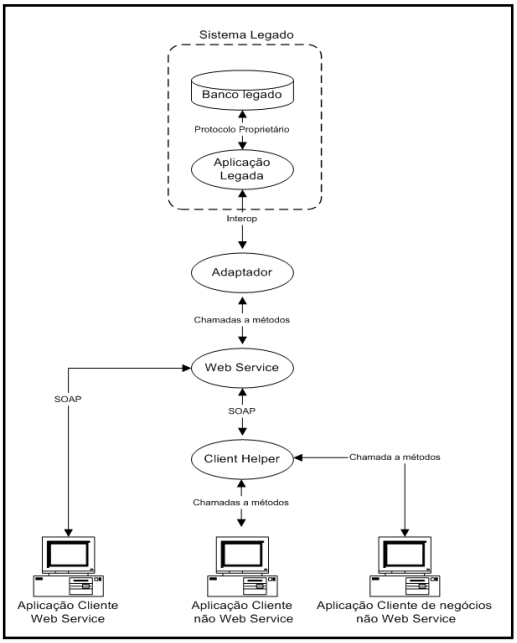

Figura 10 – Arquitetura proposta por (NAVEEN et Al). Fonte [Nave03]

5. Um Exemplo Prático

Para exemplificar, vamos criar uma interface Web simples para um determinado processo de um sistema legado operacional em uma empresa atacadista. A empresa, após análises e estudos, optou por manter seu sistema legado, desenvolvido na linguagem Clipper, executado sob o ambiente Windows. Devido a novas demandas, surgiu a necessidade de modernizar determinado processo de consulta a uma de suas informações estratégicas: as vendas anuais de cada vendedor, Tabela 1. Para disponibilizar essa informação a todos os vendedores, e de forma menos custosa possível, a Internet foi escolhida como meio.

Tabela 1 – A base legada.

Cód. Vendedor	Ano	Vendas
1	2000	100.000,00
1	2001	110.000,00
1	2002	130.000,00
1	2003	150.000,00
2	2000	80.000,00
2	2001	85.000,00
2	2002	90.000,00
2	2003	100.000,00
3	2000	140.000,00
3	2001	145.000,00
3	2002	130.000,00
3	2003	125.000,00
4	2000	120.000,00
4	2001	160.000,00
4	2002	153.000,00
4	2003	160.000,00

As informações estão armazenadas em arquivos no formato DBF (data base file), localizados em determinado diretório. A forma de integração escolhida, e menos intrusiva, é o acesso direto aos dados pertinentes contidos nestes arquivos. Como neste exemplo a informação é obtida do sistema legado, o acesso direto ao banco de dados legado é satisfatório. Por outro lado, se fosse necessário uma inclusão de dados no sistema legado, esta técnica não sería indicada. Incluindo dados diretamente na base legada poderíamos infringir regras de negócio e regras de integridade inseridas no código-fonte legado: nesse caso, outras abordagens de integração são mais apropriadas. A tecnologia escolhida para a interface Web foi JSP (Java Server Pages) e JavaBeans. JSP definirá a interface Web de entrada de dados e um JavaBean será desenvolvido para buscar os dados do sistema legado. Convém salientar que não existiu a preocupação com a segurança no acesso aos dados e critério na programação do código-fonte exemplo, por não fazerem parte do escopo e interesse deste artigo. Vamos apresentar a solução em passos:

Passo 1: Preparar o acesso à base de dados legada

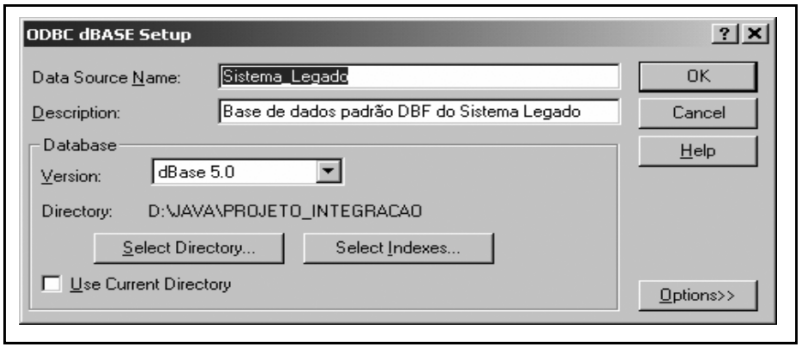

Figura 11 – Configurando o ODBC para acesso aos dados da base legada

Como o ambiente é Windows, a tecnologia ODBC (drives de acesso a dados), proprietária da empresa Microsoft Corporation, será a utilizada para o acesso aos dados legados. A Figura 11 ilustra a criação e configuração de um driver ODBC para acesso ao arquivo DBF que contém os dados que serão utilizados pela interface Web.

Passo 2: Criar a classe JavaBean responsável pela conexão à base legada

Um JavaBean será criado para prover o acesso propriamente dito aos dados da base legada. O código do mesmo é listado nas Figuras 12 e 13. No código mostrado, identificam-se alguns atributos e métodos notáveis:

Atributo **dbUrl**: Este é o "endereço do banco de dados". Sua estrutura (jdbc:odbc:Sistema_Legado) indica:

- JDBC: É a tecnologia Java (uma API) para acesso a base de dados.
- ODBC: É o subprotocolo usado. Como o ambiente é Windows, e ODBC também é um conjunto de drivers para acesso aos dados, o que se usa neste caso é uma ponte (*bridge*) JDBC<->ODBC.

· Sistema_Legado: É um identificador do banco de dados. Neste caso, é o nome dado para o driver ODBC (Figura 11).

Método **conecta**: é o método responsável por efetuar a conexão, via JDBC, ao banco de dados legado. A instrução c*lass.forName(" sun.jdbc.odbc.JdbcOdbcDriver ")* é responsável por "carregar" o driver JdbcOdbc bridge, conforme explicado anteriormente.

Método **desconecta**: irá efetuar o fechamento das conexões abertas.

Método **getValor:** é o método mais importante da classe. Ele é o responsável por retornar o valor solicitado pela interface web. Dentro deste método existe a cláusula SQL responsável por consultar a base e retornar o valor, a partir dos parâmetros obtidos da interface visual web.

```
/*
 * processa.java
 *
 * Created on 13 de Abril de 2004, 21:51
 */

/**
 *
 * @author  Herbert Laroca
 */

package beans;

import java.sql.*;

public class processa {

    private String dbUrl = "jdbc:odbc:Sistema_Legado";
    private String usuario = "";
    private String senha = "";

    private Statement s;
    private Connection c;
    private ResultSet rs;

    private String erro="";

    public void conecta() {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection c= DriverManager.getConnection(dbUrl,usuario,senha);
            s = c.createStatement();
        } catch(Exception e) {
            erro = e.toString();
        }
    }

    public void desconecta() {
      try {
          rs.close();
          s.close();
      } catch(Exception e) {
          erro = e.toString();
      }
     }
```

Figura 12 – A classe JavaBean responsável por implementar o método de busca aos dados legados.

```
public float getValor(String pvend, String panoref) {
      float valor=0;
      int vend;
      int anoref;
      if (pvend != null && panoref != null) {
         vend =  Integer.parseInt(pvend);
         anoref = Integer.parseInt(panoref);
         conecta();
         try {
            rs = s.executeQuery("Select Vendas From Vendas" +
                        " where vendedor = " + vend +
                        " and ano = " + anoref );
            rs.next();
            valor = Float.parseFloat(rs.getString(1));
         } catch(SQLException e) {
            erro =  "Erro! Dados não encontrados.";
          }
         desconecta();
}
      return valor;
   }

   public String getErro() {
      return "<B><FONT COLOR=RED> " + erro + "</FONT></B>";
   }

}
```

Figura 13 – A classe JavaBean responsável por implementar o método de busca aos dados legados (continuação).

Passo 3: Criar a interface visual que irá interagir com o usuário

O programa listado na Figura 14 implementa a interface visual web que irá interagir com o usuário. A Figura 15 ilustra a janela visualizada no browser. As instruções contidas entre as tags <% e %> é o código JSP. O restante do código é HTML. O código *<jsp:useBean id="bean1" class="beans.processa" />* registra a classe JavaBean para uso na página, e as informações digitadas pelo usuário (código do vendedor e ano de referência) são obtidas pelo comando FORM, e armazenadas nas variáveis *vend* e *anoref* . O código JSP *<%= df.format(bean1.getValor(vend, anoref)) %>* executa o método getValor (passo 2) do JavaBean e retorna o valor das vendas, que é mostrado na página. O código JSP *<%= bean1.getErro() %>* mostra eventuais erros que venham a ocorrer.

O exemplo apresentado é simples, e vários aspectos relevantes não foram abordados, mas seu acompanhamento permite uma visão prática dos conceitos apresentados neste artigo.

```
<%@page contentType="text/html"%>
<html>

<%-- <jsp:useBean id="beanInstanceName" scope="session" class="package.class" /> --%>
<%-- <jsp:getProperty name="beanInstanceName"  property="propertyName" /> --%>

<jsp:useBean id="bean1" class="beans.processa" />

<%
   String vend = request.getParameter("vend");
   String anoref = request.getParameter("anoref");
   java.text.DecimalFormat df = new java.text.DecimalFormat("###,###.00");
%>

<HEAD>
  <TITLE>Integraçao Web<->Sistema Legado</TITLE>
</HEAD>
<BODY>
 <FORM METHOD="Post" action='Processa.jsp'>
   <B><FONT SIZE=5><P ALIGN="CENTER">Integra&ccedil;&atilde;o Web &lt;&#9472;&gt;
Sistema Legado</P></FONT>
   <HR>
   <P>C&oacute;digo do Vendedor: <INPUT TYPE="TEXT" NAME="vend" >
   <P>Ano de Refer&ecirc;ncia:   <INPUT TYPE="TEXT" NAME="anoref" >
   <P>
   Valor de Vendas: <%=  df.format(bean1.getValor(vend, anoref)) %>
   <P>
   </B>
   <%= bean1.getErro() %>
   <HR>
   <INPUT TYPE="SUBMIT" VALUE="Consultar" >
   <INPUT TYPE="RESET" VALUE="Limpar">
 </FORM>
</BODY>


</html>
```

Figura 14 – O programa JSP que implementa a interface Web para o usuário

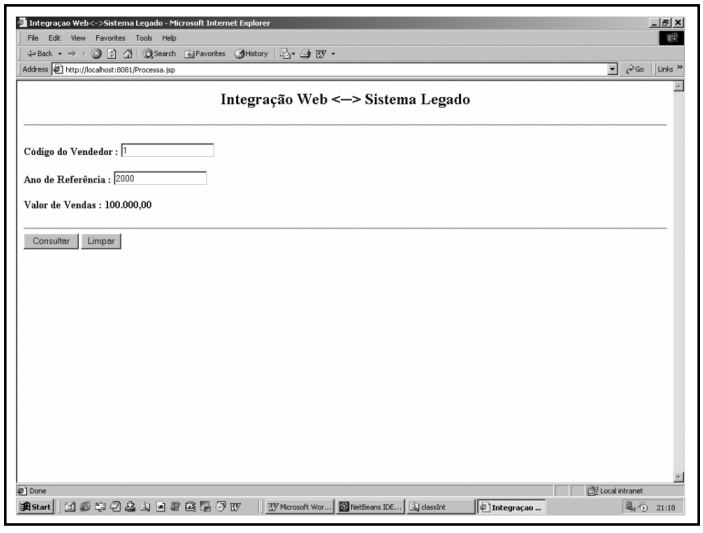

Figura 15 – Visualização da interface no browser.

6. Conclusão

Os sistemas legados existem e sempre existirão, à medida que as tecnologias vigentes tornam-se obsoletas ao longo de um tempo cada vez mais curto. O simples descarte destes sistemas incorre em prejuízos financeiros e estruturais para as organizações: o desenvolvimento de um sistema complexo envolve um grande número de recursos, e estes recursos geram gastos que precisam ser justificados e transformados em benefícios para a organização. Não obstante, a substituição do sistema legado por um novo sistema, tecnologicamente atualizado, promove também certos transtornos que devem ser avaliados, como o prazo previsto para a entrada em produção e a sua total operacionalidade, ou a garantia de que o mesmo possuirá a mesma funcionalidade fornecida pelo sistema legado antigo, no nível das regras de negócio, regras estas que nem sempre estão implícitas em manuais ou documentação.

O objetivo deste artigo foi demonstrar métodos e tecnologias existentes no presente momento para o aproveitamento dos referidos sistemas. Inicialmente, a pretensão foi fornecer informações sobre a evolução dos sistemas e uma apresentação e definição detalhada de sistemas legados, e posteriormente, a preocupação foi descrever mecanismos de avaliação de um sistema legado, para que a definição sobre a modernização ou não de um determinado sistema seja feita com grande critério. A partir da decisão de "modernizar" o sistema legado, foram apresentadas, introdutoriamente, abordagens e tecnologias para a integração do mesmo, com tecnologia ultrapassada, aos novos sistemas, com tecnologia atualizada. Não foi objetivo deste artigo apontar determinada tecnologia ou abordagem como ótima, e sim fornecer subsídios, a partir da explanação das tecnologias, para uma posterior análise da melhor abordagem e tecnologia indicada para cada caso. Não existem regras literais definidas para a escolha da abordagem ou tecnologia a ser utilizada em determinado "empacotamento" de um sistema legado. Ao longo de experiências na utilização de técnicas e metodologias deverá ser criada uma postura crítica que poderá inferir na escolha do uso das técnicas e métodos aqui apresentados.

Notoriamente, as abordagens explanadas utilizam-se de objetos distribuídos. Em conjunto com os protocolos da Internet, objetos distribuídos são a grande aposta para a integração dos sistemas, incluindo os sistemas legados. A grande ironia é que, com o advento da Internet, e a necessidade de utilização dos sistemas via esse meio, criou-se também a necessidade de integrar os sistemas corporativos antigos aos "novos" conceitos da Internet. Interfaces novas, sistemas antigos, conceitos "legados", como processamento centralizado e terminais.

Portanto, podemos concluir que não existe uma forma ou método geral que defina qual abordagem ou tecnologia utilizar. Cada caso deve ser estudado, e deve-se aproveitar o melhor de cada tecnologia disponível para se efetuar a integração. Cada sistema, complexo em sua grande maioria, irá fornecer pistas para a sua integração e o uso de várias tecnologias e abordagens tornar-se-á comum durante um projeto de integração de sistemas legados.

Várias organizações como OMG (Object Management Group), W3C (World Wide Web Consortium) e WS-I (The Web Services Interoperability Organization) estão trabalhando para criar uma padronização para a integração de sistemas

distribuídos e heterogêneos, o que irá tornar mais fácil a integração dos mesmos. A tendência verificada é o aumento crescente da facilidade de integração, com a disponibilidade cada vez maior de padrões e ferramentas para auxiliar esta tarefa. Com o passar do tempo e a maturidade das técnicas, ferramentas e padrões, os sistemas legados tendem a desaparecer, transformando-se em objetos funcionais distribuídos pelas redes das suas respectivas organizações.

7. GLOSSÁRIO

API – sigla para Application Programming Interface, ou Interface de Programação de Aplicativos. É um conjunto de rotinas e funções pré-compiladas e prontas (normalmente na forma de dll' s) que realizam uma tarefa comum. Estas interfaces foram concebidas para padronizar recursos do sistema operacional utilizados pelos aplicativos. Entre as API' s mais utilizadas, temos a API do Windows, de correio eletrônico (MAPI) e de vinculação de objetos (OLE).

Bean (Java Bean) – É um componente reutilizável de software que pode ser manipulado visualmente por qualquer ferramenta de desenvolvimento de aplicações. Possui métodos, propriedades e eventos.

Broadcast – Tradução de "radio difusão". Em uma rede de computadores, broadcast significa um aviso enviado simultaneamente para todos os micros da rede.

Clipper – Linguagem de programação criada na década de 80, muito popular no brasil nos fins da década de 80 e meados da década de 90. Sua principal característica é a facilidade de programação para o desenvolvimento de aplicações com "banco de dados" em um dialeto xbase e no formato DBF.

CODASYL – **Co**nference on **Da**ta **Sy**stems **L**anguages – Uma organização fundada em 1957 pelo Departamento de Defesa dos Estados Unidos. Sua missão foi desenvolver linguagens de programação para computadores.

DBF – **D**ata **b**ase **f**ile: Extensão de arquivos de dados criados por ferramentas xbase, como o FoxPro DOS e dbase III Plus.

HTTP – Hyper **T**ext **T**ransfer **P**rotocol: Protocolo de Transferência de Hiper Texto. Protocolo desenvolvido originalmente para transferir páginas HTML. As páginas Web são acessadas usando-se este protocolo.

IIOP – **I**nternet **I**nter-**O**RB **P**rotocol: Um protocolo desenvolvido pelo OMG - Object Management Group (OMG) para permitir soluções CORBA sobre a World Wide Web. IIOP permite que browsers e servidores web utilizem estruturas e objetos mais complexos, ao contrário do protocolo HTTP, que suporta somente texto

Middleware – Um programa que permite que dois sistemas diferentes possam se comunicar. Por exemplo, permitir que um determinado programa, capaz de acessar um determinado banco de dados, possa acessar bancos de dados em outros formatos. Outro exemplo é a possibilidade de permitir que servidores de diferentes plataformas trabalhem em conjunto.

Multithread – Capacidade de gerência e execução de várias "threads" (ver thread). Define uma característica de um sistema operacional.

ODBC – **O**pen **D**ata**B**ase **C**onnectivity: Um método de acesso a banco de dados desenvolvido pela empresa MicroSoft.

ORB – **O**bject **R**equest **B**roker: Um componente no modelo CORBA que funciona como um middleware entre clientes e servidores.

RMI – **Re**mote **M**ethod **I**nvocation: Um conjunto de protocolos desenvolvidos pela Sun que permite a comunicação entre objetos desenvolvidos na linguagem Java.

SOAP – **S**imple **O**bject **A**ccess **P**rotocol: Um protocolo baseado em XML utilizado para codificar a informação de mensagens de um Web Service antes de remete-las para a rede.

Sockets - Módulos de software que conectam os aplicativos ao protocolo de rede.

Thread – Fragmentos de código de um mesmo programa executados de forma independente, concorrentemente.

Web – "Teia" em Inglês, é um termo usado para se referir à redes de computadores. O termo surgiu devido ao formato de uma teia de aranha lembrar a disposição física de uma rede, com cabos interligando os pontos.

XML – **Ex**tensible **M**arkup **L**anguage: Uma especificação desenvolvida pelo W3C. Pode ser considerada uma extensão de HTML, permitindo a criação de tags personalizadas.

Legacy Systems and New Technologies: integration techniques and case study

KEYWORDS

Software Engineering – Information Systems – Legacy Systems – Distributed Objects

ABSTRACT

This papers aims at introducing the concept of legacy systems and the available techniques and methods for integrating them into new systems developed under new technologies, thus increasing their lifetime and usefulness. Legacy systems play a very important role in the information infrastructure of organizations, both from a strategic and economic points of view, thus making them worth of attention and development of new techniques for their seamless integration to new applications. A simple case study was developed and is presented in the paper, to demonstrate one of those integration techniques.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[Ambl98] Ambler, Scott W.. *Análise e Projeto Orientados a Objeto*, 88-371, IBPI Press, 1998
[Asma00] Asman, Paul. *Legacy Wrapping*, Federal Reserve Bank of New York, November 2000

[Brod95] Brodie, M; Stonebraker, M .. *Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach*, Morgan Kaufmann Publishers, Inc. USA, 1995.

[Chik90] Chikofsky, Elliot J. & Cross II, J.H. *Reverse Engineer and Design Recovery: A*

Taxonomy. *IEEE Software*, *7* (January 1990): 13-17.

[Come00] Comella-Dorda Santiago et al. *A Survey of Legacy System Modernization Approaches***,** Carnegie Mellon University, April 2000.

[Juric00] Juric, Matjaz B. et al. *Integrating Legacy Systems in distributed object Architecture*, Institute of Informatics, University of Maribor, 2000

[Mece99] MECELLA, Massimo; MISSIER, Paolo; MASSARI,Antonio;BATINI, Carlo.*Integration of Highly Fragmented Legacy Information Systems Through Object Modeling and Layered Wrappers* - Proceedings of the 37th Meeting of the Associazione Italiana per l'Informatica ed il Calcolo Automatico (AICA 1999), Abano Terme (PD), Italy, 1999.

[Nave03] Naveen et Al. *Web Service Facade for Legacy Applications* , Naveen Yajaman, Microsoft Corporation; Josh Brown, Implement.com; Shanmugam Subramaniam, Tony John, Narsimha Reddy, and Venkataraman R, Digitial GlobalSoft (offshore division of HP); Andrew Mason, Microsoft Corporation, 2003

[Plak99] Plakosh, Daniel; Hissam, Scott; & Wallnau, Kurt. *Into the Black Box: A Case Study in Obtaining Visibility into Commercial Software* (CMU/SEI-99-TN-010). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University. (1999).

[Somm03] Sommerville, Ian. *Engenharia de Software*, 497-513, Addison-Wesley, 2003.

[Umar97] UMAR A. *Application Reengineering – Building Web-Based Applications and Dealing With Legacy*. Prentice Hall. 1997

[Weid97] Weiderman, Nelson H.; Bergey, John K.; Smith, Dennis B.; & Tilley, Scott R. *Approaches to Legacy System Evolution* (CMU/SEI-97-TR-014). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University

[W3C03] W3C – *Web Services Architecture* – W3C Working Draft 8 August 2003

[Warr98] Ransom, Jane; Sommerville, Ian; Warren Ian. *A Method for Assessing Legacy Systems for Evolution.* Computing Dept., Lancaster University, LANCASTER LA1 4YR, UK. 1998

SOBRE OS AUTORES

HERBERT LAROCA MENDES PINTO
Analista – Desenvolvimento de Sistemas
Serviço Federal de Processamento de Dados - SERPRO.

JOSÉ LUÍS BRAGA
Professor Titular do Departamento de Informática da Universidade Federal de Viçosa
Doutor em Informática pela PUC-Rio
Pós-Doutor em Tecnologias da Informação pela University of Florida
Áreas de Interesse: ontologias, sistemas de informação cooperativos, engenharia de software, inteligência computacional