Análise comparativa entre as técnicas e metodologias de Integração de Sistemas Legados com Sistemas de Informação Atuais

Tiago Silveira de Oliveira[1], Giuliano Prado de Morais Giglio[1]

[1]Centro de Ensino Superior de Juiz de Fora (CES-JF)
Juiz de Fora, MG – Brasil

tiagojf_93@hotmail.com, giucontato@gmail.com

Abstract. *This article aims to present techniques and methodologies used in the integration of legacy systems to new information systems, addressing maintenance processes and software evolution and how the best practices related to them help in the integration process. It also presents the most used approaches to integrate these systems, highlighting in them characteristics that should be analyzed when integration becomes necessary. A comparative analysis of the approaches will be developed, based on defined and raised criteria, highlighting the characteristics analyzed in each one, in order to promote an additional contribution to IT professionals who need to evolve their legacy systems.*

Resumo. *Este artigo tem por objetivo apresentar técnicas e metodologias utilizadas na integração de sistemas legados a novos sistemas de informação, abordando os processos de manutenção e evolução software e como as melhores práticas relacionadas a eles auxiliam no processo de integração. Também serão apresentadas as abordagens mais utilizadas para realizar a integração desses sistemas, sendo destacadas nelas características que devem ser analisadas quando a integração se torna necessária. Uma análise comparativa entre as abordagens será desenvolvida, a partir de critérios definidos e levantados, destacando as características analisadas em cada uma, a fim de promover uma contribuição adicional aos profissionais de TI que necessitam de evoluir seus sistemas legados.*

1. Introdução

Muitos dos sistemas que estão presentes dentro das organizações foram desenvolvidos já algum tempo e possuem um período considerável de uso, sua tecnologia é considerada obsoleta e ultrapassada em relação as atuais, sua manutenção é complexa e custosa, além disso, são considerados críticos. Sistemas que apresentam essas características dentro de um ambiente organizacional são denominados como sistemas legados.

Ainda hoje muitos desses sistemas estão em funcionamento e são considerados como peças chaves dentro de um ambiente organizacional, ou seja, são esses sistemas que suportam os principais processos da organização. Devido a essa importância, esses sistemas são submetidos a constantes manutenções que objetivam mantê-los em funcionamento e também facilitar sua comunicação e troca de dados com outros sistemas.

Realizar essas manutenções é uma tarefa complexa e pode demandar um alto custo. Diversos fatores dificultam a realização das atividades de manutenção nesses sistemas, a falta de uma documentação do sistema para que se possa entender seus requisitos e regras de negócio é um dos principais fatores que dificultam esse processo. Há casos em que o sistema possui uma documentação, porém a mesma está desatualizada em relação ao estado atual do sistema.

A falta de *stakeholders* do sistema também é um ponto a se considerar, são eles que "conhecem" o sistema, que estiveram presentes no processo de desenvolvimento do software, sendo parte da equipe de desenvolvimento ou até mesmo funcionários da

empresa que detêm o conhecimento do negócio. Na falta da documentação do sistema ou estando a mesma desatualizada, esses *stakeholders* serão a principal referência para a equipe de manutenção.

Devido à complexidade do processo de manutenção em sistemas legados, a equipe que realiza manutenções nesses sistemas também é um ponto a se considerar, ela deve ser permanente, saber como o sistema funciona, conhecer sua estrutura profundamente e estar preparada para as constantes manutenções que serão realizadas.

Todas as atividades e processos de manutenção de software em que os sistemas legados são submetidos buscam garantir uma maior confiabilidade e manutenibilidade, e consequentemente deixar o software em um ponto em que o mesmo possa evoluir através da implementação de técnicas e processos de evolução de software.

Ao implementar os processos de evolução de software nos sistemas legados podemos torná-los independentes de tecnologia, diminuir e eliminar suas dependências, facilitar e simplificar sua manutenção, tornando-os sistemas manuteníveis. Utilizando os conceitos de evolução de software temos por principal objetivo evitar a degradação do sistema e consequentemente seu descarte, garantindo que o mesmo tenha uma vida mais longa, obtendo uma estrutura mais robusta capaz de acompanhar as mudanças do ambiente no qual está inserido (Furtado, 2007).

Com o surgimento de novas tecnologias e o aumento da diversidade e quantidade de sistemas dentro das organizações, é de interesse delas que seus sistemas se interajam, de maneira que informações, processos e funcionalidades sejam compartilhadas. Com o apoio dos processos de evolução e manutenção de software e de técnicas de integração, que através da utilização de conceitos de orientação a objetos, uso de *web services,* entre outras tecnologias utilizadas como forma de integração e comunicação de sistemas em diferentes linguagens e plataformas, esse compartilhamento se torna possível.

Com esse panorama, o seguinte trabalho propõe a realização de uma pesquisa exploratória com base na literatura descrevendo as técnicas e metodologias de integração utilizadas para integrar sistemas legados a novos sistemas, tecnologias e novas abordagens existentes, processos de manutenção e evolução de software utilizados para garantir a manutenabilidade de sistemas legados, e os desafios atuais de sua utilização.

Com essa base, essa pesquisa é justificada pelo fato de que em trabalhos já realizados apenas tecnologias obsoletas são apresentadas, não sendo feita nenhuma comparação ou destacando suas características baseando-se em algum padrão ou norma, sendo que atualmente novas tecnologias e abordagens já existem e estão sendo adotadas nos processos de integração entre sistemas legados a novos sistemas, e sua análise e comparação em relação às tecnologias antigas ainda não foram exploradas.

2. Evolução dos Softwares e Processo de Manutenção para Garantia da Manutenabilidade

Evoluir um software é um conceito amplo e que pode envolver vários aspectos, que podem se originar da necessidade de torná-lo adaptável ao ambiente no qual está inserido e para isso utilizar de técnicas de manutenção, ou realizar sua reengenharia, visando evitar sua degradação e torná-lo mais robusto.

As atividades de evolução de software podem ser divididas em três categorias: manutenção, modernização e substituição. Conforme é ilustrado na figura 1, as atividades de evolução são aplicadas as diferentes fases do ciclo de vida de um sistema. A linha pontilhada representa o crescimento das necessidades de um negócio, enquanto a linha sólida representa a funcionalidade fornecida pelo sistema de informação (Pinto and Braga, 2005).

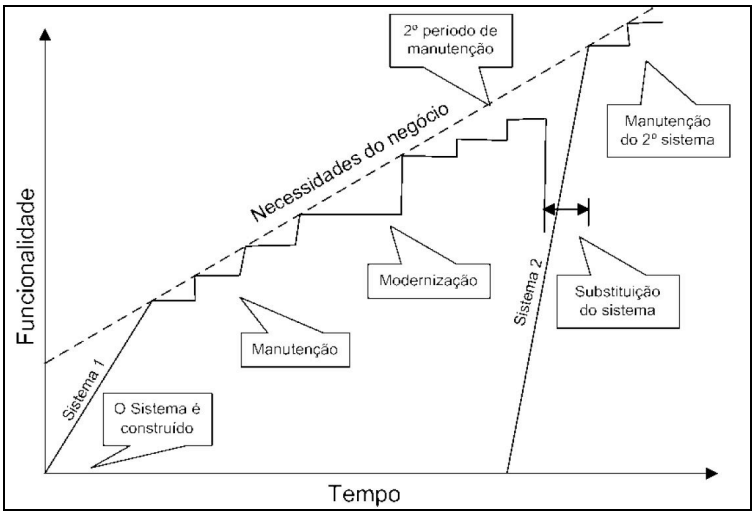

Figura 1. Ciclo de vida de um sistema de informação
Fonte: [Pinto and Braga, 2005]

A manutenção é um processo incremental e iterativo em que pequenas modificações são efetuadas no sistema. Essas modificações vão desde correções que objetivam a identificação e remoção de falhas à pequenas melhorias no sistema, sendo que grandes mudanças estruturais nunca são realizadas.

A modernização envolve mudanças maiores que a manutenção, mas conserva grande parte do sistema. Essas mudanças vão desde implementação de novas funcionalidades à mudança de arquitetura do sistema.

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

Ao realizar atividades de evolução de software em um sistema, devemos considera-lo sob duas perspectivas distintas em relação ao ambiente ao qual ele está inserido: a perspectiva de negócio, onde é avaliado o valor do sistema para a organização; e a perspectiva de sistema, onde é avaliada a qualidade do software de aplicação. Através dessa avaliação, podemos classificá-los em quatro grupos de sistemas: baixa qualidade e baixo valor de negócio, baixa qualidade e alto valor de negócio, alta qualidade e baixo valor de negócio e alta qualidade e alto valor de negócio.

Sistemas classificados como sendo de baixa qualidade e baixo valor de negócio, são candidatos a serem descartados; por outro lado, sistemas de baixa qualidade e alto valor de negócio, são candidatos a transformação ou substituição.

Sistemas de alta qualidade e baixo valor de negócio, são aqueles em que sua substituição não compensa e sua manutenção pode ser continuada, ou até mesmo podem ser substituídos. Já os sistemas de alta qualidade e alto valor de negócio devem ser mantidos em operação e que devido a sua alta qualidade, sua transformação ou substituição não é necessária.

As atividades de manutenção, modernização e substituição podem abranger vários aspectos dentro da evolução de software, realizá-las em sistemas complexos e críticos exige um profundo conhecimento de sua estrutura, por isso é necessário que o sistema seja avaliado corretamente em relação ao seu ambiente e valor para a organização, para que se evite prejuízos futuros e gastos desnecessários para a empresa.

2.1. Manutenabilidade de Software

O conceito de manutenabilidade de software pode ser definido como a facilidade em que um produto de software pode ser compreendido, corrigido, adaptado e/ou aperfeiçoado. Para determinar a influência da manutenabilidade de software, podemos classificar alguns fatores de impacto (Silva et al, 2010):

A arquitetura, que pode ser considerada como sendo um investimento que aprimora a manutenabilidade. A divisão da aplicação em componentes e de componentes em classes, torna a unidade de código bem menor em relação a outros componentes, facilitando o esforço de modificação e diminuindo consideravelmente a inclusão de erros durante as modificações.

A tecnologia, que deve permitir um grau satisfatório de manutenabilidade. Sistemas que utilizam programação orientada a objetos apresentam um grau positivo de manutenabilidade, devido ao fato de que as manutenções mais localizadas levarem a menor degradação do código.

A documentação, que pode ser considerada um ponto de grande importância devido ao fator tempo. O tempo gasto para compreender o produto antes de modificá-lo é maior quando a documentação está indisponível ou desatualizada.

E a compreensibilidade do software. Grande parte do tempo de manutenção é utilizado para entender a documentação e lógica do produto de software. Esses fatores de impacto fazem parte de todo ciclo de vida do software. A interdependência desses fatores irá determinar o grau de manutenabilidade de um software.

Como grande parte dos principais sistemas dentro de um ambiente organizacional são legados, integrá-los a novas tecnologias e sistemas pode ser considerada uma tarefa árdua, mas torna-se possível com a implementação de processos de manutenção que visam garantir uma maior confiabilidade e manutenabilidade, e técnicas de evolução que possibilitam a sua evolução e comunicação com outros sistemas dentro de um mesmo ambiente.

2.1.1. Cenário Atual da Evolução e Manutenção de Software

Além dos processos e técnicas já conhecidas no ciclo de manutenção e evolução, atualmente existem ferramentas que apoiam e automatizam esses processos. Tendo em vista que os processos de manutenção e evolução de um sistema legado é trabalhoso e podem demandar tempo e custo, o uso dessas ferramentas pode ser considerado vital para realização desses processos, podendo diminuir o tempo gasto em seu entendimento e até mesmo evitar custos desnecessários para a organização.

No contexto da integração de sistemas legados, essas ferramentas podem auxiliar na identificação de relacionamento entre componentes, sendo úteis em casos em que um componente é um serviço baseado em SOA. Ao realizar esse tipo de integração, é necessário que se tenha conhecimento de tudo relacionado a esse componente, seus limites, o que pode ou não ser alterado e principalmente seus relacionamentos, por isso o uso dessas ferramentas torna-se importante, não apenas para os processos de manutenção e evolução, mas também para o de integração.

3. Abordagens para integração de sistemas legados

Com o passar do tempo muitos dos sistemas ainda em uso dentro das organizações tornam-se obsoletos, com isso e objetivando atender seus negócios e manter sua competitividade de mercado, as organizações buscam novas soluções que utilizam tecnologias de software mais recente. Durante a implantação dessas soluções é notado que nem todos os sistemas antigos podem ser substituídos, isso ocorre porque alguns deles atendem necessidades bastante especificas da organização, além de que sua substituição demandaria gastos, podendo até ser prejudicial para os negócios organização em diversos sentidos.

Com base nesse cenário, é do interesse da organização que os novos sistemas compartilhem e troquem informações com seus sistemas legados, gerando assim uma necessidade de integração entre eles.

Existem várias abordagens de integração entre sistemas e, não sendo possível apontar qual delas é a melhor, sua utilização se dá de acordo com os objetivos e necessidades da organização e, levando em consideração, os sistemas e ambiente no qual estão inseridos. Com o objetivo de apresentar e mostrar como essas abordagens são implementadas, nesse capítulo serão apresentadas abordagens: que utilizam a tecnologia

de *web services*, SOA (*Services Oriented Architecture*), EAI (*Enterprise Application Integration*), EJB (*Enterprise JavaBeans*) e *Cloud Computing*.

3.1. Web Services

No ano de 2000, a W3C (*World Wide Web Consortium*) aceitou a submissão do SOAP (*Simple Object Access Protocol*), um formato de mensagens baseado em XML que estabeleceu uma estrutura de transmissão para a comunicação entre serviços via HTTP. No decorrer do ano seguinte a W3C publicou a especificação WSDL (*Web Service Description Language*), uma nova implementação do XML que forneceu uma linguagem para descrever a interface dos *web services*. Posteriormente a especificação UDDI (*Universal Description, Discovery and Integration*) proporcionando um mecanismo padrão para a descoberta dinâmica de descrições de serviços, estabelecendo assim a primeira geração de *web services* (Erl, 2009).

A tecnologia *web services* propõe incrementar interoperabilidade entre sistemas utilizando padrões abertos, não importando linguagem utilizada, plataforma de hardware ou software, ou até mesmo localização geográfica dos sistemas integrados (Zavalik, 2004). A especificação de padrões dos *web services* pode ser implementada por diversos fabricantes, podendo ser caracterizados pelos seguintes aspectos:

- Descrição - os *web services* descrevem suas funcionalidades e atributos, para que outras aplicações possam descobrir como usá-los.
- Publicação - os *web services* são registrados em um repositório que contém informações sobre como conectar e usá-los.
- Descoberta - quando um *web service* é localizado, uma aplicação remota pode invocar o serviço.
- Retorno de resposta - quando um *web service* é invocado, os resultados são retornados para a aplicação solicitante.

3.1.1. Arquitetura Web Service

Utilizando tecnologias já existentes na internet, o HTTP, como método de transporte, e o XML como formato de mensagens, os *web services* dependem apenas de três padrões que são fundamentais e que possibilitam a comunicação, independentemente da plataforma em que as aplicações estão sendo executadas ou linguagem de programação. São eles:

SOAP: Permite a comunicação entre aplicações ou serviços.

WSDL: Descreve o *web service*, permitindo que outros *web services* saibam como acessá-lo, o que mandar como entrada e esperar como saída.

UDDI: Permite que *web services* publiquem documentos WSDL para outros *web services*. Também fornecem especificações sobre formatos de entrada e saída de dados, modelos de segurança e protocolos.

3.1.2. Exemplo de integração utilizando *Web Services*

Afim de demonstrar a utilização da tecnologia *web service* será apresentado um sistema da prefeitura de Juiz de Fora, DIM (Dispensação Individualizada de Medicamentos), que é responsável pela liberação de medicamentos nas UAPS (Unidades de Atenção Primária a Saúde), onde houve a necessidade de uma integração com o sistema de uma empresa terceirizada que distribui medicamentos para todas as unidades (Peracci, 2015).

O sistema DIM implantado em 2012 tem como função a liberação de medicamentos nas farmácias das unidades em cada bairro, junto com o controle e ajuste de estoque. Mensalmente, um pedido dos medicamentos é realizado pelo sistema da empresa terceirizada e entregues nas unidades os medicamentos referentes a solicitação. Com isso, o usuário responsável pela liberação no sistema tinha a função dispensar o medicamento e além disso realizar a entrada no estoque do sistema, manualmente, e um medicamento por vez, de aproximadamente 120 medicamentos, demandando tempo e desgaste do funcionário (Peracci, 2015).

Diante dessas dificuldades, em 2013 a solução encontrada foi a utilização do *web service* para que pudesse haver uma comunicação entre os dois sistemas e utilizando como referência a tecnologia SOAP, melhorando a entrada de material no estoque do DIM (Peracci, 2015).

O DIM foi desenvolvido utilizando PHP como linguagem de programação e *MySQL* para armazenamento de dados (Peracci, 2015).

Para um melhor entendimento de como a integração entre os sistemas foi realizada, a seguir ilustramos a mesma em 5 passos (Peracci, 2015):

1. A farmacêutica da UAPS acessa o sistema da empresa terceirizada;
2. Em seguida, solicita a reposição dos medicamentos, onde são informados os medicamentos e a quantidade necessária;
3. É feita uma verificação no estoque da empresa terceirizada e a aprovação do pedido é feita;
4. Um serviço Web é disparado, contendo as informações solicitadas referentes aos medicamentos para o banco de dados e em seguida, para o estoque DIM;
5. Com os medicamentos já no estoque, a farmacêutica pode realizar a liberação dos medicamentos.

A figura 2 ilustrada o fluxograma de integração, sendo Sistema X responsável pela distribuição/logística dos medicamentos (Peracci, 2015).

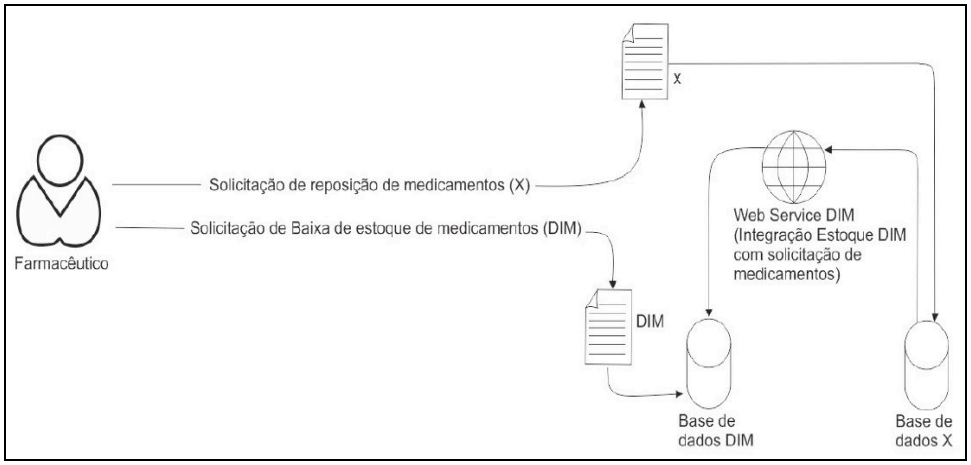

Figura 2. Fluxograma de integração
Fonte: [Peracci, 2015]

Utilizando a tecnologia *web services* para realizar a integração dos dois sistemas houve uma diminuição no tempo de espera do paciente para a entrega do seu medicamento, menor desgaste para o funcionário que realiza a dispensação, aperfeiçoamento do sistema, disponibilização dos serviços para futuras empresas que venham assumir a distribuição dos medicamentos e agilidade nos processos (Peracci, 2015).

3.2. SOA (*Services Oriented Architecture*)

Não se sabe ao certo quando SOA foi criada, alguns dizem que a arquitetura foi desenvolvida em 1994, por um analista do grupo Gartner chamado Alenxander Pasik. Outros dizem que os primeiros indícios e discussões foram por volta do ano de 2000, de estudos da Microsoft e IBM a tecnologia de *web service* (Oliveira, 2013).

Durante o processo de integração, podem ocorrer situações indesejadas como: incapacidade de redução no tempo dos processos por limitações de integração; lentidão na identificação de oportunidades e ameaças pela falta de uma visão integrada de

diferentes sistemas. Como resposta a essa situação, a arquitetura orientada a serviços é vista como um caminho a ser seguido (Serman, 2010).

SOA designa um conjunto de arquiteturas e padrões de desenvolvimento de software focados na disponibilização e consumo de serviços entre aplicativos em uma rede. Cada componente da arquitetura SOA representa um serviço, ou uma funcionalidade especifica do sistema. "Um serviço pode consumir outro serviço, utilizando uma funcionalidade já implementada, permitindo a criação de serviços mais simples até serviços mais complexos" (Silva and Gonçalves, 2012).

Os aplicativos consumidores utilizam o serviço não se preocupando com sua implementação, apenas acessando uma interface comum que esconde os detalhes do modelo de dados e do processamento realizado.

A tecnologia de *web services* é parte importante da arquitetura SOA, ela permite que serviços sejam desenvolvidos, publicados, descobertos e consumidos.

3.2.1. Serviços

Serviços são componentes de software construídos de forma que possam ser facilmente vinculados a outros componentes de software. "Como objetivo principal, o conceito de serviços tem uma visão de que é possível definir partes do código de software em porções significativas o suficiente para serem compartilhadas e reutilizadas em diversas áreas da empresa, assim algumas tarefas passam a ser automatizadas" (Klein, 2010).

3.2.2. Exemplo de integração utilizando SOA

Enquanto as organizações privadas veem os sistemas de informação como ferramentas fundamentais para garantir competitividade e investem em sistemas eficientes, integrados e interoperáveis, as organizações públicas apresentam uma grande dificuldade em adotá-los. No entanto, a necessidade de uma melhor gerência dos sistemas de informação começa a despertar nos gestores públicos a busca de sistemas integrados, que proporcionem benefícios em termos de agilidade e transparência (Cunha; Júnior; Dornelas, 2008).

As instituições públicas da rede federal, tendo o CEFET-AL como um caso particular, são geridas por diferentes sistemas de informação, que além de complexos devem trocar informações entre si. O CEFET-AL ainda não possui o controle efetivo da sua informação, uma vez que trabalha com sistemas isolados, sem integração e sem perspectivas de fornecerem informações que auxiliem o processo de tomada de decisões. Os problemas apontados referentes ao sistema foram avaliados no sentido de buscar soluções adequadas com foco nos objetivos estratégicos da instituição (Cunha; Júnior; Dornelas, 2008).

Como solução, foi proposto a implementação de um protótipo de integração orientada a serviços para os sistemas de informação do CEFET-AL, com o objetivo de obter uma operação mais eficaz dos processos de negócio da instituição e diminuir a inconsistência e replicação de dados. Os problemas levantados foram avaliados para buscar soluções adequadas com foco nos objetivos estratégicos da instituição. Como resultado chegou-se aos serviços listados conforme o quadro 1 (Cunha; Júnior; Dornelas, 2008):

Nome do Serviço	*Descrição*	*Sistema Provedor*
Levantar Distribuição de Material por Setor	Obter localização dos bens nos setores.	Patrimônio
Levantar Cota de Investimento por Setor	Obter valor investido de materiais de consumo por setor.	Almoxarifado
Levantar Relação Aluno Investimento	Calcular o investimento por aluno na instituição.	Acadêmico e Orçamento e Estatística
Levantar Dados Orçamentários por Fonte de Recursos	Obter os valores orçamentários por rubrica oriundos das fontes de recursos da instituição.	Orçamento e Estatística
Levantar Capacitação de Servidores	Identificar formação, cursos e treinamentos realizados pelos servidores.	Cadastro e Lotação
Controlar Diárias	Levantar diárias concedidas aos servidores.	Financeiro

Quadro 1. Lista de serviços da arquitetura de integração
Fonte: [Cunha; Júnior; Dornelas, 2008]

Neste é apresentada a arquitetura de integração baseada em SOA proposta para o CEFET-AL. "É ilustrado um conjunto de serviços, sistemas e base de dados que se comunicam de acordo com o fluxo de negócio da instituição. Além dos serviços identificados e listados, outros serviços podem ser adicionados a arquitetura" (Cunha; Júnior; Dornelas, 2008).

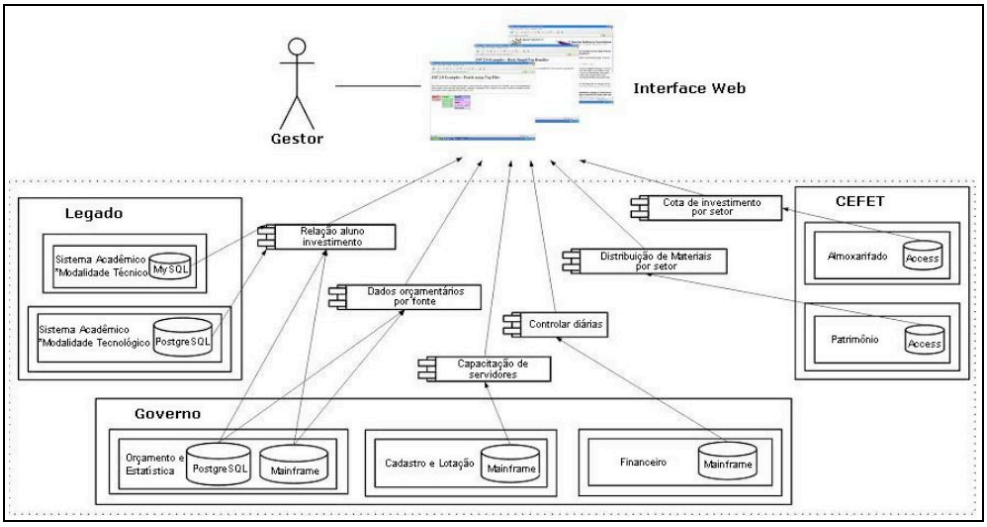

Figura 3. Lista de serviços da arquitetura de integração
Fonte: [Cunha; Júnior; Dornelas, 2008]

Conforme apresentado na figura 3, os serviços são agrupados em três categorias (Cunha; Júnior; Dornelas, 2008): Legado, Governo e CEFET. O primeiro grupo corresponde aos sistemas de terceiros, onde o CEFET-AL não possui documentação ou código-fonte. O segundo grupo engloba serviços de acesso aos sistemas do governo. E o terceiro grupo corresponde aos serviços que acessam os sistemas desenvolvidos no CEFET. Todos os serviços apresentados são provedores, sendo a interface web o sistema consumidor principal.

A comunicação entre os serviços da arquitetura e os sistemas integráveis é realizada através do acesso direto a suas bases de dados. A comunicação entre serviços é feita pela rede, utilizando a tecnologia *web services* (Cunha; Júnior; Dornelas, 2008).

3.3. EAI (*Enterprise Application Integration*)

Enterprise Application Integration (EAI) existe como um termo técnico desde 2000, mas o problema central que ela tenta resolver é muito antigo. Em outras palavras,

EAI é uma abordagem que visa fornecer interoperabilidade entre múltiplos sistemas dispares que compõem uma infraestrutura coorporativa típica.

Um fator crítico é observar como sistemas e aplicações legadas se encaixam na nova visão concebida, analisando-se as possibilidades de sua utilização integral ou parcial, assim como seu relacionamento com os novos aplicativos adicionados a nova plataforma tecnológica da empresa. "A utilização de EAI é fundamental para empresas que possuem sistemas e aplicativos heterogêneos e não integrados, que estão em diferentes linguagens, plataformas e padronizações de comunicação entre si" (Oliveira, 2003).

Nesse contexto, EAI utiliza-se de tecnologias genéricas, como *drivers* de integração, *middlewares*, gerenciadores de transações, *brokers* orientados a objetos e camadas de servidores web (Oliveira, 2003).

3.3.1. Modelos de Integração

O principal objetivo do EAI é permitir que uma organização integre diversas aplicações de forma rápida, fácil e reduzindo o acoplamento Oliveira (2003):

> *As organizações devem compreender tanto os processos de negócio quanto os dados, e devem selecionar quais deles requerem integração. O processo de integração ser visto por várias dimensões, destacando-se: os níveis de dados, interface das aplicações, métodos e interface do usuário (Oliveira, 2003).*

O nível de dados consiste em processos, técnicas e tecnologias que permitem mover, transportar dados entre diferentes fontes e destinos, tratando os dados, fazendo as atualizações necessárias e mantendo a integridade dos mesmos (Oliveira, 2003).

O nível de interface da aplicação está relacionado com a utilização das interfaces das aplicações e pacotes de sistemas. Utilizando as interfaces, pode-se empacotar muitas aplicações permitindo o compartilhamento lógico e de informação. Para realizar a integração, é necessário o uso destas interfaces para que se tenha acesso tanto dos dados quanto os processos, extraindo informação, substituindo, colocando no formato adequado para aplicação de destino e transmitindo (Oliveira, 2003).

O nível dos métodos consiste no compartilhamento da lógica de negócios, por exemplo, um mesmo método pode ser acessado por uma quantidade considerável de aplicações e essas aplicações podem utilizar outros métodos sem precisar reescrever um método para cada aplicação (Oliveira, 2003).

O nível de interface do usuário consiste na criação de um ponto comum de integração para os usuários, uma interface gráfica. Aplicações que funcionam no ambiente *mainframe* podem ser exibidas de forma visual ao usuário (Oliveira, 2003).

3.3.2. Exemplo de integração utilizando EAI

Como exemplo prático, será ilustrada a realização de uma integração EAI a nível de dados. Com o objetivo de exemplificar, será criada uma interface *web* simples para determinado processo de um sistema legado operacional em uma empresa atacadista (Pinto and Braga, 2005).

Após análises e estudos, foi optado por manter seu sistema legado, desenvolvido na linguagem *Clipper* e 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. Para disponibilizar essa informação a todos os vendedores, e com menos custos, a internet foi escolhida como meio (Pinto and Braga, 2005).

As informações estão armazenadas em arquivos no formato DBF (*data base file*), localizados em determinado diretório. A tecnologia escolhida para a interface web foi JSP (*Java Server Pages*) e *Java Beans*. JSP definirá a interface web de entrada de dados e um *Java Bean* será desenvolvido para buscar os dados do sistema legado (Pinto and Braga, 2005).

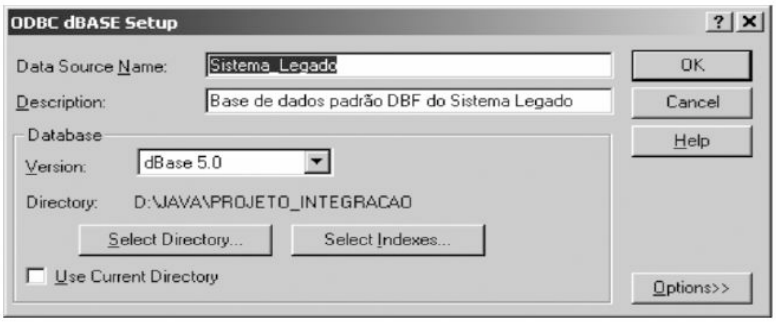

Figura 4. Configuração ODBC para acesso à base legada
Fonte: [Pinto and Braga, 2005]

Conforme a figura 4, é realizada a configuração do *driver* ODBC para acesso ao arquivo DBF que contém os dados que serão utilizados pela interface *web*. Em um *Java Bean* foram implementadas funções de conexão e desconexão que utilizam o *driver* ODBC como meio de acesso aos dados da base legada. Os dados são recebidos por uma página JSP e apresentados via *browser* para o usuário final (Pinto and Braga, 2005).

O exemplo apresentado é simples e aborda a integração a nível de dados de forma direta através do acesso a base legada utilizando a tecnologia ODBC. Nesse exemplo, a tecnologia ODBC funciona como um ponto de conexão entre a base legada (arquivo DBF) e a classe *Java*, implementada para gerenciar o acesso aos dados e exibi-los ao usuário através da página JSP via *browser*.

3.4. EJB (*Enterprise Java Beans*)

A primeira versão do EJB, a versão EJB 1.0, foi lançada em março de 1998 na conferência de desenvolvedores *Java* chamada *Java One* 1998. Ao longo das versões sucessoras, EJB 1.1, 2.0 e 2.1, diversas alterações foram feitas e funcionalidades foram incluídas, como a compatibilidade com protocolos CORBA e o suporte aos *web services*.

EJBs são componentes do lado do servidor, utilizados para implementar a camada funcional de uma aplicação, seu modelo de programação os torna menos complexos, altamente reutilizáveis, escaláveis e integráveis com outras tecnologias, sistemas e modelos de dados. O componente responsável por realizar o encapsulamento da lógica de negócio é o *Session Bean*, sendo ele classificado em três tipos: *Stateful*, *Stateless* e *Singleton* (Junior and Bavaresco, 2016).

Em um *session bean stateful*, as variáveis são mantidas até o final da sessão, em contrapartida no *session bean stateless* nenhum estado de conversação com o cliente é mantido. O *session bean singleton* é instanciado apenas uma vez e utilizado em casos em que uma única instância é compartilhada e acessada pelos clientes (Junior and Bavaresco, 2016).

3.4.1. Exemplo de integração utilizando EJB

O primeiro passo para empacotar um sistema legado utilizando EJB é separar a interface do sistema em módulos consistindo em unidades lógicas. O próximo passo é construir um único ponto de contato para o sistema legado, podendo esse ponto ser implementado como um *bean*, *adapter*, ou um *service broker*, componente externo ao servidor EJB. A etapa final é implementar um componente empacotador para cada módulo do sistema legado (Pinto and Braga, 2005).

Conforme a figura 5, o sistema legado foi dividido em unidades lógicas distintas, representadas pelas funções 1 e 2. Para cada unidade lógica existente no sistema legado, é criada sua respectiva *bean* no servidor EJB que será responsável pelo seu encapsulamento, as *beans* 1 e 2 representam respectivamente as funções 1 e 2 no sistema legado. O ponto de conexão entre o sistema legado e o servidor EJB pode ser

implementado como uma *bean*, *adapter*, no servidor EJB ou um componente externo ao servidor e o sistema legado. As requisições ao sistema legado são realizadas pelas *beans* no servidor EJB através do ponto de conexão entre eles.

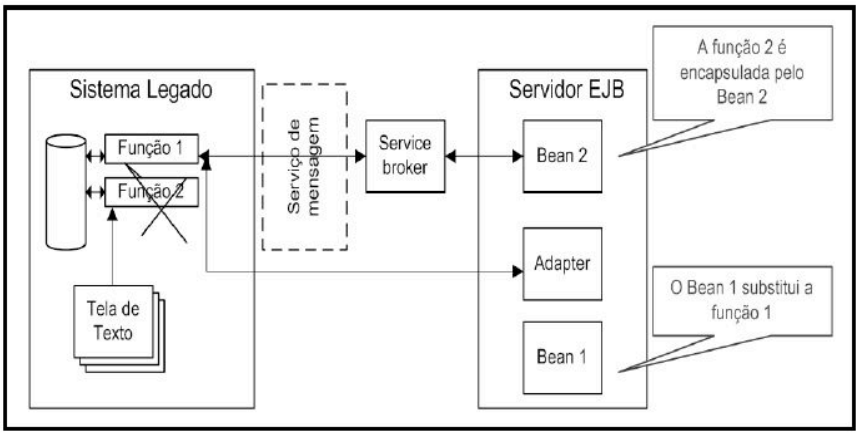

Figura 5. Empacotamento da lógica legada utilizando EJB
Fonte: [Pinto and Braga, 2005]

3.5. Cloud Computing

Relatos históricos apontam que a ideia inicial sobre *cloud computing* surgiu no início da década de 1960, quando o cientista da computação norte-americano, John McCarthy, inventor do termo "inteligência artificial", defendeu a proposta de computação por tempo compartilhado (*time-sharing*). Esse modelo foi apresentado durante um discurso no *Massachusetts Institute of Technology* (MIT), nos EUA, em 1961. Ele sugeriu a criação da computação como serviço de utilidade pública (*utility computing*).

Em 1962, Joseph Carl Robnett Licklider, do MIT, já falava sobre a criação de uma rede intergaláctica de computadores. E em 1969, Leonard Kleinrock, cientista norte-americano que chefiava o *Advanced Research Projects Agency Network* (Arpanet), órgão que criou a internet, endossou o conceito de *utility computing* de McCarthy.

De acordo com Muhlbeier (2011):

> *Com o Cloud Computing, os aplicativos não precisam estar instalados ou armazenados em seu computador. Ao fornecedor da aplicação cabem todas as tarefas de desenvolvimento, armazenamento, manutenção, atualização, backup, escalonamento, etc. O usuário não precisa se preocupar com nada disso, basta acessar e utilizar. Todos os seus materiais e documentos estarão disponíveis em qualquer ambiente, independente de sistema operacional ou hardware onde a aplicação esteja rodando (Muhlbeier; Oliveira; Mozzaquatro, 2011).*

Cloud computing pode ser considerada como um serviço computacional oferecido através da internet de acordo com a necessidade do cliente por um provedor especializado. Essa tecnologia oferece recursos de modo econômico, escalável e flexível, sendo acessível e atrativo para empresas que desejam reduzir seus custos e ter possibilidade de transferi-los ao desenvolvimento de seu *core business* (Junior and Pires, 2010).

3.5.1. Exemplo de integração utilizando Cloud Computing

Existe grande interesse no investimento em *cloud* pelas grandes companhias, esse interesse se justifica pelo desenvolvimento de novas ou adaptadas aplicações e sistemas, como exemplo o sistema *EyeOS* (Muhlbeier; Oliveira; Mozzaquatro, 2011).

Sistema *EyeOS*

Ferramenta *open source* que começou a ser desenvolvida em 2004. Tem objetivo de suportar uma grande variedade de aplicações *web*. O sistema cumpre tarefas

básicas como edição de textos, execução de arquivos mp3 e vídeos, navegação, jogos, comunicação entre usuários e vários outros aplicativos e mini aplicativos (*widgets*) que devem ser instalados e configurados pelo administrador do sistema. O *EyeOS* possui seu próprio sistema de arquivos virtual, o *EyeFiles*, que possibilita a criação, *upload* e armazenamento de arquivos no próprio sistema. O sistema permite acesso via *browser* a todas as funcionalidades, conforme a figura 6 (Muhlbeier; Oliveira; Mozzaquatro, 2011).

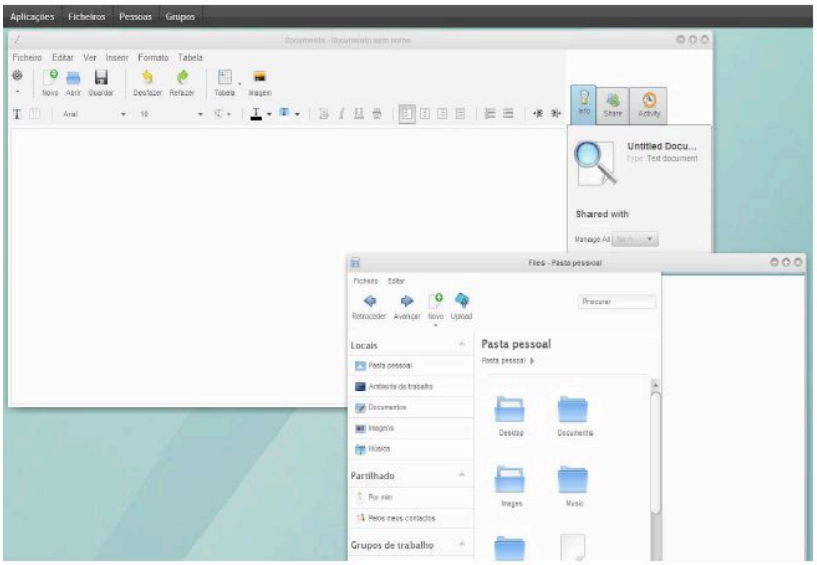

Figura 6. Sistema virtual de arquivos, *EyeFiles*.
***EyeOS* versão 2.5.**
Fonte: [Muhlbeier; Oliveira; Mozzaquatro, 2011]

Para realizar a integração entre o AVA *Moodle* ao sistema *EyeOS*, foi necessária a instalação do banco de dados *MySQL* e o *Apache Server*. O AVA *Moodle* integrado ao *EyeOS* 2.5, disponibiliza um *link* que acessará o AVA *Moodle,* além de documentos compartilhados entre os integrantes (Muhlbeier; Oliveira; Mozzaquatro, 2011).

Essas são algumas das abordagens utilizadas quando a integração entre um sistema legado e novos sistemas se torna necessária. Para cada situação, deve-se analisar qual dessas abordagens se encaixa melhor e irá atender de forma satisfatória a necessidade de integração. Não sendo possível dizer quando e qual delas deve ser utilizada, sua escolha como forma de integração se dá de acordo com o cenário em que esses sistemas estão inseridos.

4. Comparação entre as abordagens de integração

Como citado anteriormente, a utilização de cada abordagem se dá de acordo com a necessidade da organização e/ou ao cenário em que os sistemas envolvidos estão inseridos. Com esse panorama, esse capítulo propõe realizar um estudo comparativo entre as abordagens apresentadas anteriormente, destacando interoperabilidade e coexistência, como sub-características de compatibilidade, e modificabilidade como sub-característica de manutenabilidade, presentes na norma ISO/IEC 25010, sendo esses critérios escolhidos dentre os demais pelo fato de que em um processo de integração, é esperado que os sistemas se comuniquem e troquem informações, que eles operem em um mesmo ambiente e quando necessário, sua manutenção não seja custosa e não demande muito tempo para realizá-la.

Essas questões estão diretamente relacionadas as características de interoperabilidade, coexistência e modificabilidade, junto a elas também serão abordados custo, implementação e desempenho, também podendo ser considerados pontos de interesse no processo de integração, estando a característica de custo

relacionada ao investimento necessário para utilização da abordagem de integração, a implementação, relacionada a questões pertinentes ao processo de integração que utiliza a abordagem, e desempenho, que está relacionado a performance obtida com a utilização da abordagem.

De acordo com Morais (2014):

> *A norma ISO/IEC 25010 é parte do projeto SQuaRE (Software product Quality Requirements and Evaluation), um conjunto de normas técnicas que estabelece padrões para qualidade e que inclui gerência, modelo, medição, requisitos e avaliação de qualidade para qualquer produto de software. Seu modelo de qualidade para produto de software é composto pelas dimensões qualidade do produto e qualidade em uso (Morais and Costa, 2014).*

Para escolher uma tecnologia que melhor se adapta a necessidade da organização, ambiente e sistemas a serem integrados, é necessário que essas características sejam bem avaliadas antes que o processo de integração se inicie, com a finalidade de evitar complicações durante ou após o processo.

4.1. Análise e comparação das abordagens de integração

Ao realizar a integração entre sistemas legados à novos sistemas, é esperado que haja interoperabilidade entre eles, ou seja, que tanto o sistema legado quanto os novos sistemas se comuniquem, trocando e compartilhando informações. Que eles coexistam, não havendo nada que impossibilite a interação entre ambos dentro de um ambiente comum. E que sua modificação não seja morosa, não havendo dificuldades para modificar e/ou adaptá-los.

Essas questões estão relacionadas às características de interoperabilidade, coexistência e modificabilidade, que podem ser consideradas as mais importantes durante o processo de integração. Elas devem ser analisadas levando em conta fatores como: tipo de sistemas que serão integrados, ambiente no qual esses sistemas serão implantados e principalmente a vida útil esperada dessa integração, se ela será um projeto de curto ou a longo prazo.

Com esse objetivo, as abordagens apresentadas anteriormente foram analisadas com base nessas características junto a custo, implementação e desempenho (conforme os quadros de 2 a 7):

Modificabilidade	
Web Services	A capacidade de modificação dos web services pode ser problemática, podendo afetar no custo de manutenção do serviço oferecido.
SOA	Serviços podem ser reutilizados em outras aplicações, isolando sua estrutura, as mudanças são feitas com maior facilidade.
EAI	Por utilizar tecnologia proprietária e adaptadores especializados para integrar sistemas, esses sistemas podem ser tornar legados e sem capacidade de reagir a mudança.
EJB	Sua facilidade de manutenção está diretamente ligada à sua implementação.
Cloud Computing	A capacidade de se modificar é uma qualidade embutida em ambientes *cloud*.

Quadro 2. Análise das abordagens em relação a modificabilidade
Fonte: Do Autor

Na abordagem *web services* há uma certa dificuldade quando se torna necessário realizar modificações, isso ocorre pelo fato de que os *web services* são utilizados nos mais variados tipos de sistemas e possuam implementações distintas. Em

contrapartida devido a sua facilidade de modificação, na abordagem SOA os serviços possuem um alto índice de reutilização, isolando sua estrutura e aplicando alterações necessárias, eles podem ser reutilizados em outras aplicações. Já na abordagem EAI existe certa dificuldade na realização de modificações, devido a utilização de tecnologias proprietárias que resultam na dependência de seu fornecedor. Em EJB modificar e/ou realizar manutenção está diretamente relacionado a sua implementação, ou seja, sendo bem implementada, não haverá grandes dificuldades para modificar sistemas que utilizam essa abordagem. Na abordagem *cloud computing* a capacidade de modificação está presente no ambiente *cloud,* não havendo muitas dificuldades na realização de modificações.

Coexistência	
Web Services	A única exigência é que a comunicação siga os padrões estabelecidos pelas partes.
SOA	Por ser baseado no uso de padrões, sua integração com outros serviços, aplicativos e sistemas legados é facilitada.
EAI	Utilizando soluções proprietárias de diferentes fornecedores, a perspectiva de integração terá uma complexidade elevada.
EJB	É classificado como *framework* de integração de *middleware*, por prover suporte à comunicação remota entre componentes e como *framework* de infraestrutura, por tratar questões típicas de infraestrutura entre as aplicações.
Cloud Computing	Utilizam protocolos já existentes na internet, sua interação com outros sistemas é facilitada.

Quadro 3. Análise das abordagens em relação à coexistência
Fonte: Do Autor

Nas abordagens *web services* e SOA a coexistência com outros sistemas é facilitada, pelo fato de ambas utilizarem padrões e tecnologias já existentes na internet. Na abordagem EAI, por existirem soluções proprietárias de diferentes fornecedores, utilizando linguagens proprietárias entre outros fatores, a coexistência entre sistemas é dificultada. Com EJB, a coexistência entre os componentes e os serviços das aplicações é gerenciada pelo servidor EJB, onde é implementado o ambiente de execução para os mesmos. Em *cloud computing* a coexistência com outras aplicações é facilitada pelo fato de que a abordagem utiliza a internet como meio de comunicação.

Interoperabilidade	
Web Services	Os *web services* permitem ligar qualquer tipo de sistema, independentemente das plataformas e linguagens de programação utilizadas.
SOA	Os serviços são disponibilizados independentemente da plataforma e tecnologia.
EAI	Fornecem um processo de integração centralizado e monolítico que geram desvantagens, como: linguagem proprietária no processo de integração; plataforma especifica para cada nova plataforma; dependência de um único ponto.
EJB	O servidor EJB implementa o ambiente de execução para o componente e gerencia serviços comuns entre as aplicações, como segurança, transações, estados, compartilhamento de recursos, persistência automática e chamada remota.

Cloud Computing	A integração de software ocorre automaticamente e organicamente, gerando esforço mínimo para customização e integração.

Quadro 4. Análise das abordagens em relação à interoperabilidade
Fonte: Do Autor

Como na coexistência, nas abordagens *web services* e SOA a interoperabilidade com outros sistemas é facilitada, por ambas utilizarem padrões e tecnologias já existentes na internet. Em EAI, as soluções proprietárias, como na coexistência, também é um fator que dificulta sua interoperabilidade com outros sistemas. Em EJB, a interoperabilidade entre os componentes e os serviços das aplicações também é gerenciada pelo servidor EJB. Na abordagem *cloud computing* a interoperabilidade com outras aplicações também é facilitada, pela utilização da internet como meio de comunicação.

Implementação	
Web Services	O tempo de implementação de sistemas com a utilização de *web services* é mais reduzido.
SOA	Uma implementação bem-sucedida depende de um planejamento cuidadoso da arquitetura de negócios, um conjunto de boas práticas, governança de processos, serviços e pessoas, metodologia de desenvolvimento centralizado e o envolvimento de todos que estão envolvidos nos processos.
EAI	Complexidade da arquitetura a ser integrada, falta de mão-de-obra familiarizada com as tecnologias EAI e aumento de segurança de acordo com o nível de integração, podem ser considerados riscos a sua implementação.
EJB	O desenvolvedor apenas se preocupa com a camada de regra de negócios, deixando para o EJB questões que fujam desse contexto.
Cloud Computing	A aplicação *cloud* é implementada como um serviço, entregando o que ele foi designado a fazer de forma que isso seja mais importante que a tecnologia utilizada para o disponibilizar.

Quadro 5. Análise das abordagens em relação à implementação
Fonte: Do Autor

Por não haver necessidade de desenvolver aplicações a partir do zero a implementação da abordagem *web services* é realizada facilmente, não sendo necessário muito investimento. Em contrapartida, para implementar SOA, é necessário planejamento, governança de processo e envolvimento de todas as pessoas dentro da organização. Na abordagem EAI, como em todo projeto, existem pontos a se considerar, principalmente quando se trata em mão-de-obra especializada com as tecnologias EAI, essa questão entre outras, podem causar risco a sua implementação. Em EJB, como em sua coexistência e interoperabilidade, sua implementação também é gerenciada pelo servidor EJB. Na abordagem *cloud computing*, sua implementação se dá através da utilização da internet como meio de comunicação entre as aplicações, que são implementadas como serviços projetados de acordo com a necessidade do cliente.

Desempenho	
Web Services	O desempenho dos *web services* está relacionado ao protocolo SOAP, que por sua vez possui limitações que os afetam diretamente em nível do desenho das aplicações, chamadas remotas, características da rede, armazenamento e

	processamento dos documentos.
SOA	A performance depende do servidor onde o serviço está publicado, como também da rede.
EAI	A redução no número de camadas por onde os dados têm de passar até chegar ao seu destino, promove uma melhor performance durante o processo de troca de dados entre aplicações.
EJB	Alguns problemas podem afetar diretamente no desempenho do EJB, como: alterações feitas em outros sistemas ou diretamente no banco são refletidas imediatamente pelas *beans*; para fazer buscas, é necessário mais de uma *query* para trazer todos os dados.
Cloud Computing	As instâncias *cloud* podem ser adicionadas instantaneamente para aumentar a performance.

Quadro 6. Análise das abordagens em relação ao desempenho
Fonte: Do Autor

Na abordagem *web services*, seu desempenho diretamente relacionado as limitações presentes no protocolo SOAP. Em SOA, ele está relacionado ao servidor onde o serviço está publicado. Em relação à desempenho, a abordagem EAI promove uma melhor performance durante as trocas de dados entre as aplicações. Já em EJB, ele pode ser afetado quando alterações são realizadas diretamente no sistema ou no banco de dados, refletindo imediatamente nas *beans*. Na abordagem *cloud computing*, devido a capacidade das instancias *cloud* serem adicionadas instantaneamente quando necessário, o desempenho *cloud* pode ser aumentado quando necessário.

Custo	
Web Services	Os *web services* utilizam protocolos e infraestrutura web já existentes, demandando pouco investimento das partes interessadas.
SOA	Implantar SOA implica um alto investimento inicial. Uma análise custo/benefício deve ser realizada antes de migrar a arquitetura para uma perspectiva SOA.
EAI	Altos custos para se implantar o sistema devido a arquitetura, desenvolvimento da integração e as operações do ambiente integrado.
EJB	Altos custos em relação ao desenvolvimento, e custos para o cliente.
Cloud Computing	Baixos custos devido a eliminação de investimentos em servidores exclusivos para a aplicação, custos associados com o desenvolvimento e manutenção da infraestrutura.

Quadro 7. Análise das abordagens em relação ao custo
Fonte: Do Autor

Por utilizar tecnologias já existentes na internet, não é necessário muito investimento na utilização da abordagem *web services*. Já em SOA, para sua implantação, um alto investimento inicial é demandado, sendo necessário a realização de uma análise custo/benefício antes de migrar para essa arquitetura. Na abordagem EAI, altos custos decorrentes dos processos de implantação, desenvolvimento e operações são demandados. Em EJB, sua utilização demanda altos custos, devido a fatores relacionados ao desenvolvimento e custo para o cliente. Por ser utilizada como serviço, podendo os mesmos serem contratados de acordo com a necessidade, os custos de utilização da abordagem *cloud computing* são baixos.

5. Considerações Finais e Conclusão

Sistemas legados sempre existirão dentro das organizações e ao mesmo tempo a necessidade para que eles se comuniquem e troquem informações com novos sistemas, internos ou externos a empresa, também existirá. Visando garantir a interoperabilidade entre esses sistemas, integrá-los pode ser uma solução, mas apenas realizar essa integração pode não ser o suficiente se o sistema legado não estiver preparado para esse processo, ter um certo nível de maturidade e principalmente ser manutenível. Nessas questões, os processos de manutenção e evolução de software podem ser empregados com a finalidade de se alcançar uma melhor manutenabilidade e maturidade na estrutura dos sistemas.

Antes que esses processos se iniciem, a empresa deve avaliar o valor que o sistema possui para os seus negócios, e sendo esse um sistema de alto valor, as melhores práticas e ferramentas que auxiliam tanto na manutenção quanto na evolução de software devem ser utilizadas, assegurando sua qualidade e garantindo o mesmo seja evoluído a um ponto em que integrá-lo a outros sistemas não seja moroso a empresa.

Sendo a integração de sistemas legados o principal objetivo desse artigo, as abordagens apresentadas são as mais utilizadas para integrar esses sistemas, de acordo com a pesquisa exploratória realizada na literatura, e foram analisadas e comparadas com base nas principais características almejadas durante o processo de integração.

Para cada projeto, ambiente e sistemas a serem integrados, existe uma abordagem que melhor se adequa ao seu contexto de integração, portanto, não é possível dizer qual delas é a melhor, mas sim, indicar qual delas se encaixa em uma determinada situação. Dentre as abordagens apresentadas a *web services* é a mais utilizada nos processos de integração por sua implementação não ser complexa e utilizar tecnologias já existentes na internet, sendo utilizadas para integrar sistemas de pequeno a grande porte.

A abordagem SOA pode ser indicada nos casos em um serviço precisa acessar dados de um determinado sistema, não havendo a necessidade de realizar uma integração completa a esse sistema.

A abordagem EAI pode ser aplicada em casos de integrações mais localizadas, como uma integração a nível de dados, onde apenas o acesso a base de dados de um determinado sistema é necessário. Nesse tipo de integração, *middlewares*, *brokers* orientados a objetos, entre outros componentes são utilizados como meios de comunicação entre os sistemas a serem integrados.

A abordagem EJB dentre todas as apresentadas pode ser considerada a mais complexa, sendo indicada para sistemas de grande porte. Sua escolha como forma de integração deve ser feita em projetos de longa duração, e se mal utilizada, pode causar problemas em sua manutenção, performance e portabilidade, devido ao fato de sua implementação estar relacionada a essas questões. Como apresentado, sistemas mal desenvolvidos e implementados, posteriormente podem causar problemas em processos de manutenção e evolução, na abordagem EJB esses processos estão diretamente relacionados a sua implementação.

A abordagem *cloud computing* é a mais nova dentre as apresentadas em termos de integração e é adotada como um serviço, onde são escolhidos os serviços que melhor atende ao contexto de integração, sendo esses serviços disponibilizados de forma gratuita ou paga, em casos em que a expansão desses serviços seja necessária ou até mesmo contratá-los de fornecedores específicos, devido ao processo de integração.

As abordagens apresentadas não são absolutas, podendo ser aprimoradas à medida em que são utilizadas, podendo até atuarem simultaneamente. Dentre elas, a abordagem *cloud* é a mais recente, podendo ser uma das mais exploradas e utilizadas futuramente, pelo fato de possibilitar o acesso as informações em qualquer lugar e nos mais variados tipos de dispositivos. Em contrapartida, a abordagem EAI, por utilizar tecnologias proprietárias, linguagem própria, demandar altos custos de utilização e causar grande dependência de seu fornecedor, pode futuramente cair em desuso, sendo a ascensão da tecnologia *cloud* um fator que contribua para isso. As abordagens *web services*, SOA e EJB ainda continuam sendo as mais utilizadas atualmente e evoluem de acordo com as necessidades do mercado, diferente da abordagem EAI, futuramente sua utilização não deve cair em desuso com certa facilidade, sendo possível sua adaptação a novas tecnologias.

Como trabalho futuro, pode-se realizar a implementação das técnicas apresentadas a partir de um sistema legado, com o objetivo de atestá-las em termos de implementação.

Outro trabalho, se daria através da comparação das características destacadas em cada abordagem sendo apresentadas de forma estatística, com base na integração entre um sistema legado com um sistema que utiliza tecnologias mais recentes, sendo essa integração realizada utilizando cada uma das abordagens apresentadas e pontuando em valores estatísticos os resultados alcançados em relação a cada característica.

Em relação a essas características, cabe a organização avaliá-las durante o processo de integração, se o investimento na adoção da abordagem irá compensar a curto ou longo prazo, levando em consideração o sistema, ambiente e principalmente a finalidade dessa integração.

Referências

CUNHA, Mônica X. C; JÚNIOR, Marcilio F. Souza; DORNELAS, Jairo Simião. **O uso da arquitetura SOA como estratégia de integração de sistemas de informação em instituição pública de ensino**. Simpósio de Excelência em Gestão e Tecnologia, 2008.

ERL, Thomas. Web Services. 2009. **DevMedia**. Disponível em: <https://www.devmedia.com.br/web-services/2873>. Acesso em: 22 jun. 2018.

FURTADO, Thiago Bellotti. **Evolução de Software: Fundamentos, Processos e Aplicação**. 2007. Monografia – Centro de Ensino Superior de Juiz de Fora, Juiz de Fora - MG, 2007.

JUNIOR, Alvondi R. Lima; BAVARESCO, Jorge L. Boeira. **Estudo Comparativo de Tecnologias da Plataforma Java EE para Desenvolvimento de Sistemas Web**. 2016. Monografia – Instituto Federal Sul-Rio-Grandense, Passo Fundo, 2016.

JUNIOR, João B. Camargo; PIRES, Sílvio R. I. **Sistemas Integrados de Gestão ERP e *Cloud Computing*: Características, Vantagens e Desafios**. Simpósio de Administração da Produção, Logística e Operações Internacionais. São Paulo, 2010.

KLEIN, Vítor. O que é arquitetura orientada a serviços (SOA)?. 2010. **Vítor Alberto Klein's Blog**. Disponível em: <https://vitoralbertoklein.wordpress.com/2010/02/20/o-que-e-arquitetura-orientada-a-servicos-soa/>. Acesso em: 22 jun. 2018.

MORAIS, Rinaldo Macedo; COSTA, André Lucirton. Um modelo para avaliação de sistemas de informação do SUS de abrangência nacional: o processo de seleção e estruturação de indicadores. **Revista de Adm. Pública**. v. 48, n. 3, p. 767 – 93, 2014.

MUHLBEIER, Andreia R. Kessler; OLIVEIRA, Leander Cordeiro; MOZZAQUATRO, Patrícia Mariotto. **Integração entre AVA *Moodle* e o Sistema *EyeOS*: A Relevância**

de sua Utilização. 2011. Monografia – Universidade de Cruz Alta, Cruz Alta – RS, 2011.

OLIVEIRA, Edson Mendes. Vantagens e Desvantagens de SOA. 2013. **DevMedia**. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=27437>. Acesso em: 22 jun. 2018.

OLIVEIRA, Nívea Menezes. **Web Services e J2EE: Uma Solução para Integração de Sistemas em Ambientes Heterogêneos**. 2003. Tese – Centro de Tecnologia e Geociências, Universidade Federal de Pernambuco, Recife – PE, 2003.

PERACCI, Rodolfo Fernandes. **Utilizando Web Service para Integração de Sistemas Um Estudo de Caso: Prefeitura de Juiz de Fora**. 2015. Monografia – Centro de Ensino Superior de Juiz de Fora, Juiz de Fora – MG, 2015.

PINTO, Herbert L. Mendes; BRAGA, José Luís. Sistemas Legados e as Novas Tecnologias: técnicas de integração e estudo de caso. **Informática Pública**, v. 7, p. 47 – 69, 2005.

SERMAN, Daniel Valente. Orientação a Projetos: Uma Proposta de Desenvolvimento de uma Arquitetura Orientada a Serviços. **Revista de Gestão da Tecnologia e Sistemas de Informação**. v. 7, n. 03, p. 619 – 638, 2010.

SILVA, Ricardo Frenedoso; GONÇALVES, Pablo Rodrigo. Web Services – Uma Análise Comparativa. **Revista das Faculdades Integradas Claretianas**. n. 5, 2012.

SILVA, Ronan Assumpção; MATOS, Simone Nasser; SOUZA, João U. Furquim; MOURA, Louisi Francis. **A Manutenção de Software nas Empresas**. Congresso Internacional de Administração. Ponta Grossa – PR, 20 set. 2010.

ZAVALIK, Claudimir. **Integração de Sistemas de Informação através de Web Services**. 2004. Tese – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.