ALEXANDRE DEL NERO
DANILO ELIAS DA SILVA
RICARDO MOREIRA MUNIZ

SISTEMA DE GESTÃO EMPRESARIAL PARA PEQUENAS EMPRESAS QUE UTILIZA UMA ARQUITETURA SOA E NUVENS DE SERVIÇO

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia de Computação.

São Paulo
2010

ALEXANDRE DEL NERO
DANILO ELIAS DA SILVA
RICARDO MOREIRA MUNIZ

SISTEMA DE GESTÃO EMPRESARIAL PARA PEQUENAS EMPRESAS QUE UTILIZA UMA ARQUITETURA SOA E NUVENS DE SERVIÇO

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia de Computação.

Área de concentração:
Engenharia da Computação

Orientador:
Prof. Jorge Risco Becerra

São Paulo
2010

Este trabalho é dedicado aos nossos pais e amigos, motivadores majoritários para a conclusão de nosso curso.

AGRADECIMENTOS

Agradecimentos especiais para os professores que, por meio do ensino, viabilizaram a conclusão deste curso, em especial, ao professor Jorge Risco, que balizou o término desta dissertação com valiosos conselhos.

Agradeço ao professor Marcos José Garino e à inspetora de alunos Maria Lúcia Joaquim Groto pela fundamentação que culminou com o diploma de graduação, sem os quais o mesmo não seria possível.

RESUMO

O presente trabalho visa abordar sistemas de baixo custo para gestão empresarial por meio de uma arquitetura orientada a serviços em nuvens, utilizando padrões web.

Visando melhorar a estrutura de TI dessas empresas, é proposta uma arquitetura de sistemas flexível e orientada a serviços (SOA) que utiliza como prova de conceito um sistema para gestão empresarial (ERP) baseadas em padrões web e disponibilização de serviços em uma nuvem de aplicação (PaaS).

Palavra-chaves: SOA, ERP

ABSTRACT

The current work discuss about low-cost systems for business management through a service-oriented architecture in cloud computing and using Web standards.

To improve the IT infrastructure of these companies, we propose a flexible system architecture and service oriented (SOA) that uses as a proof of concept a system for enterprise resource planning (ERP) based on web standards and providing services in a PaaS (Plataform as a Service) cloud.

Key-words: SOA, ERP

Lista de Figuras

Sumário

1 Introdução

"Before I refuse to take your questions, I have an opening statement."

Ronald Reagan

Um elemento importante quanto ao desempenho das empresas é a incorporação de novas tecnologias aos seus processos produtivos, uma vez que a inovação e a tecnologia possibilitam meios de produzir os mesmos produtos, ou mesmo novos, com eficiência. Neste contexto, as pequenas empresas comumente figuram como participantes secundários, pois, comumente, não possuem recursos ou mesmo o embasamento técnico para implantar e manter soluções que, de outro modo, poderiam ser diferenciais quanto a sua, tão difícil, e no Brasil onerosa, permanência no mercado.

Visando solucionar esse impasse o presente trabalho propõe uma arquitetura comum de sistemas integrados de gestão ou ERP (Enterprise Resource Planning) ao mesmo tempo simplificada, de modo a facilitar a personalização e manutenção por pequenas empresas, flexível, de modo a abranger variadas personalizações, e escalável, suportando, por fim, a agregação de funcionalidades em seus processos de negócios como constituintes do processo natural de crescimento das mesmas.

Para tanto, uma extensa análise da literatura e dos produtos existentes (*estado da arte*) foi feita objetivando uma proposta atual e que visa, por fim, contribuir, mesmo que pouco, para esta area tão nova, ou muito específica, que é a automatização dos processos de negócio de pequenas empresas em uma solução genérica e extensível.

1.1 Motivação

As pequenas empresas prestam contribuições singulares à economia brasileira. Das quase 7 milhões de empresas formais, as micro e pequenas empresas (MPEs) representam 98,9% - 60% do emprego formal no país [26] [4].

Mesmo assim, sem recursos, estrutura e pessoal qualificado não conseguem processar e obter em tempo hábil as informações para se adaptar e acompanhar às mutações na mesma velocidade que as grandes, sofrendo conseqüentemente da punição advinda dos maus resultados, não raramente causa do encerramento das suas atividades. Por outro lado, precisam se estruturar para poderem competir com as grandes organizações, ou mesmo com outras empresas de mesmo porte.

São inerentemente frágeis, pois, já no contexto do TI, pela sua própria natureza, as MPEs são menos propensas a sobreviver a implementações falhas de sistemas organizacionais [18], necessitando de sistemas flexíveis, a fim de alcançar a competitividade necessária, que abram espaço para a inserção de processos de negócio que, comumente, são muito específicos. Soma-se a isso

a necessidade intrínseca de simplicidade, tenha em vista que as barreiras iniciais de uso de um sistema cooperam para o seu fracasso - o que é especialmente verdadeiro para as MPEs.

Há, por fim, a necessidade de baixos custos para estas, que refletem em escalabilidade a nível de infraestrutura, pois supre as necessidades de uma grande gama de pequenas empresas, pulverizando custos individuais. Escalabilidade, esta, justificada também pelo processo natural de crescimento das empresas, que demanda uma reengenharia constante de seus processos de negócio.

1.2 Objetivo

No presente trabalho é proposta uma arquitetura flexível orientada a serviços (SOA) de sistemas de gestão empresarial (ERP) baseada em padrões abertos e nuvens de serviço. Destaca-se, também, que esta solução foi pensada como sendo de pequeno porte, tendo como foco as micro e pequenas empresas.

1.3 Organização do documento

O Capítulo 2 visa fundamentar a escolha do escopo deste trabalho, justificando-o no contexto empresarial, bem como, nas pequenas e médias empresas, por meio de dados ou referências científicas. No Capítulo 3 são introduzidos os conceitos de serviço, tanto na sua disponibilização como seu consumo, culminando com tendências para sua agregação conforme alguns especialistas no assunto e uma análise breve do estado da arte, por meio de comparações com sistemas existentes. São descritas, a seguir, no Capítulo 4, a proposta e o projeto realizado em torno da mesma, visando comprovar sua viabilidade e fazendo alguns testes com o mesmo objetivo, e, por fim, esta dissertação conclui-se com as lições aprendidas, trabalhos futuros e um sumário do documento no Capítulo 5. Os Capítulos 6 e 7 destinam-se aos anexos e bibliografia, respectivamente.

2 Sistemas de informação organizacionais

2.1 As pequenas empresas no Brasil e sua caracterização

As micro e pequenas empresas são um dos principais sustentáculos da economia brasileira, quer pela sua enorme capacidade geradora de empregos, quer pelo infindável número de estabelecimentos desconcentrados geograficamente.

Em termos estatísticos, das quase 7 milhões de empresas formais, 93,8% são microempresas, sendo que o conjunto micro e pequenas empresas (MPEs) representa 98,9%, chegando a representar 60% do emprego formal no país [26] [4].

Ressalte-se sua crescente importância nos últimos anos, dado que, cada vez mais, as MPEs vêm experimentando um ambiente mais propício a novos negócios [13], fruto dos benefícios auferidos pela linha de incentivos legais às mesmas, adotadas nos últimos anos, iniciada pelo *Estatuto Nacional da Microempresa e Empresa de Pequeno Porte*, de 1998 [30].

Este trouxe consigo uma série de benefícios que vêm sofrendo constantes complementações no sentido de simplificar e auxiliar processos de MPEs. Temos, por exemplo, a *Lei Geral da Micro e Pequena Empresa*, de 2006, que resultou em incentivos e desburocratização, e foi seguida pelo *Simples Nacional*[1], em 2007, visando a redução de carga tributária para as mesmas e impulsionando seu desenvolvimento e sua formalização.

Mas, afinal, quem são as MPEs? Legalmente, embasados nestes dispositivos legais, as definimos como sendo:

- **Microempresas**: A pessoa jurídica que aufira, em cada ano-calendário, receita bruta igual ou inferior a R$ 240.000,00 (duzentos e quarenta mil reais) [30].
- **Empresas de Pequeno Porte**:A pessoa jurídica, ou a ela equiparada, que aufira, em cada ano-calendário, receita bruta superior a R$ 240.000,00 (duzentos e quarenta mil reais) e igual ou inferior a R$ 2.400.000,00 (dois milhões e quatrocentos mil reais) [30].

2.2 Computação em nuvens no contexto da terceirização

Quando se fala em sistemas de gestão empresarial, cujo processo de instalação e configuração costuma levar anos [33], o papel das empresas de consultoria que auxiliaram na implantação quase nunca desaparece inteiramente, de modo que a organização cliente frequentemente se torna dependente da primeira, seja por meio de contratos de serviço formais (*Service Level Agreement*), ou pelo conhecimento íntimo do sistema de gestão - especialmente verdadeiro para pequenas empresas, por apresentarem riscos muito maiores nessa implantação.

[1] Sistema Integrado de Pagamento de Impostos e Contribuições das Microempresas e Empresas de Pequeno Porte

Dito isso, a terceirização (*outsourcing*) de tais sistemas tem sido vista como uma opção atraente por muitas organizações devido a crença de que os fornecedores deste tipo de serviço podem alcançar economias de escala e especialização, pois seu único negócio é o processamento de informações, deixando-as livres para atender o mercado no seu negócio principal [36].

Tais fornecedores, por sua vez, a fim de serem bem sucedidos, devem prover um nível de serviço maior do que os clientes poderiam alcançar por si mesmos, bem como devem fazê-lo de modo a tornar viável a contratação do serviço para a empresa contratante - comumente por meio de preços baixos.

Um modelo de terceirização particularmente interessante na área de tecnologia que vêm ganhando muita popularidade e aceitação em tempos recentes é o modelo que se apoia em computação em nuvens (*cloud computing*), onde o sistema está instalado e é configurado em um espaço físico sob o domínio do provedor de serviço, sendo portanto alheio a organização cliente, que passa, então, a atuar fazendo backups diários e atualizações periodicas em conformidade com as novas versões lançadas pela organização responsável pelo desenvolvimento da solução de gestão em uso, mantendo um canal de suporte ativo com o cliente e cobrando taxas fixas mensais pelos serviços prestados.

As razões para a adoção desse modelo incluem [36]:

- Busca por eficiência organizacional
- A liberdade de não precisar dispor de recursos para recrutar e treinar pessoal especializados em manter infra-estrutura e aplicações
- Otimização do fluxo de caixa simplificada por ter gastos fixos em TI
- Foco no negócio principal da empresa

Existem diferentes tipos de serviços oferecidos na computação em nuvem e a empresa contratante deve verificar quais são aqueles que melhor atendem suas necessidades. Os serviços podem ser classificados como:

- **Infrastructure as a Service (IaaS)**: Tem como foco disponibilizar infra-estrutura como serviço. Oferece aos usuários recursos de hardware, sistema operacional e outros softwares complementares sob demanda. Isso significa que o usuário pode especificar quanto ele necessita de capacidade de armazenamento, de memória virtual, de processamento e essa capacidade computacional ser disponibilizada como serviço e pode ser alterada a qualquer momento. O usuário também tem a capacidade de selecionar qual sistema operacional deseja utilizar dentre aqueles oferecidos pela empresa que foi contratada e se quer utilizar alguns softwares básicos como firewalls. Esse tipo de serviço é utilizado, normalmente, por clientes

que necessitam de computadores para determinada necessidade, mas no querem adquirir os produtos, pois são onerosos para a empresa e também existem os gastos para manter uma infra-estrutura computacional. A escalabilidade é obtida através de virtualização e distribuição das máquinas virtuais através dos servidores na nuvem. O Amazon's EC2 e o Gogrid são exemplos desses serviços. [23] [11]

- **Platform as a Service (PaaS)**: Tem como objetivo oferecer um ambiente de desenvolvimento ou framework para que os clientes desenvolvam, testem, disponibilizem e mantenham aplicativos através da nuvem. Os clientes no precisam se preocupar em manter o ambiente nem ter gastos com ele, pois patches e upgrades são feitos automaticamente. Os custos são baseados nos recursos que o cliente utiliza. Alguns exemplos de serviços desse tipo são o Azure da Microsoft, o AppEngine da Google e o Force.com da SalesForce.com. [10]

- **Software as a Service (SaaS)**: É a disponibilização de um software como serviço. Ele acessado pelo cliente, tradicionalmente, pela Internet. No contexto de computação em nuvem, o software desenvolvido para este fim utiliza a infra-estrutura da nuvem. Ou seja, se apóia nos outros tipos de serviço, como pode ser visto na figura.

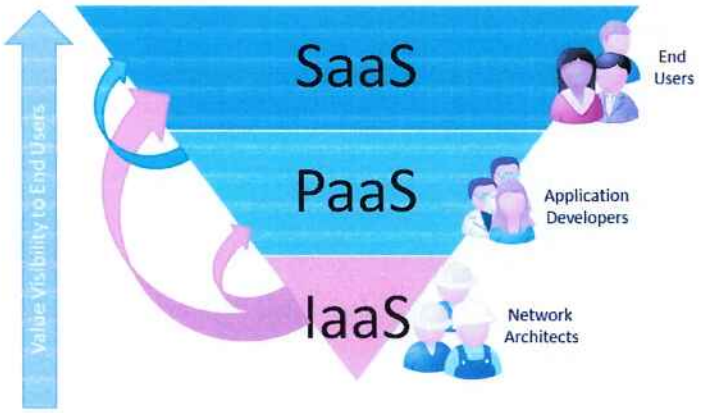

Figura 1: Tipos de serviços presentes na computação em nuvem e suas interdependências [34]

Como pode-se observar nessa organização em camadas, a camada IaaS responsável pela infra-estrutura e a escalabilidade que suporta o software, a camada PaaS pelo desenvolvimento do software e a camada SaaS por disponibilizar para os usuários o software. Então, a disponibilidade e segurança do software são altamente dependentes da qualidade dos serviços das outras camadas. O conceito de software como serviço é muito comum atual-

mente. Podemos encontrar pela Internet inúmeros serviços desse tipo. Por exemplo, o Google utiliza-se sua enorme infra-estrutura na nuvem e disponibiliza serviços como o Gmail, o Google Docs e o Google Maps.

2.3 Enterprise Resource Planning

Os *Sistemas de Gestão Empresarial* (*Enterprise Resource Planning*, ou *ERP*) são tipicamente definidos como sendo sistemas modulares orientados ao negócio que visam facilitar a gerência e o uso eficiente de recursos (materiais, recursos humanos, financeiros, etc) provendo uma solução total e integrada para as necessidades de processamento de informação das organizações. Visam facilitar o fluxo de informação entre funcões do negócio dentro da organização e gerênciar as conexões com stakeholders externos. Tais sistemas possuem o potencial de melhorar processos de negócio, análise de dados, reduzir inventários e auxiliar tomadas de decisão, sendo, muitas vezes, o maior investimento em TI feito por uma organização [15] - chegando a somas da ordem de US$15 milhões por ano, com um tempo médio de 23 meses para sua implementação efetiva [33].

É, pois, de importância capital que a escolha de uma solução que vise a integração dos processos de negócio de uma organização seja bem fundamentada e que os benefícios provenientes de uma possível solução sejam comprovadamente concretos e necessários para a sua viabilização.

Nesse sentido, esta seção visa apresentar dados concretos sobre a inserção de sistemas de gestão no contexto organizacional e apresentar os resultados advindos desta inserção, destacando-se, entretanto, que tal como ocorre com os objetivos e processos organizacionais, as soluções nesse sentido são muito específicas, inexistindo soluções *genéricas* - havendo, porém, soluções *flexíveis*.

2.3.1 História

Os ERPs têm sido, desde a década de 90, um dos principais focos de atenção da utilização da Tecnologia de Informação (TI) pelas organizações. Adquiridos na forma de pacotes comerciais de softwares, utilizam uma base única de dados, que permite a integração, em tempo real, de todos os sistemas de informações transacionais e dos processos de negócios da organização como um todo e não apenas de departamentos isolados como nos pacotes tradicionais.

Tais sistemas representam o ultimo estágio na evolução e expansão de técnicas de produção e controle nas empresas. Desde os sistemas de planejamento de requisitos materiais (MRP), passando pelos sistemas de planejamento de requsitos de capacidade (CRP) aos sistemas de planejamento de recursos na manufatura (MRP II). Sistemas MRP II começaram a evoluir em sistemas ERP até que, em meados de 1988, a Dow Chemical Company adquiriu seu primeiro sistema ERP da SAP AG da Alemanha. Durante o período de 1988 a 1994 os termos MRPII e ERP foram usados indistintamente [18].

O termo "Enterprise Resource Planning"foi usado pela primeira vez para descrever sistemas voltados ao controle de recursos internos pelo Gartner Group de Stanford, Connecticut, USA.

Esses sistemas se evidenciaram em 1994, quando a SAP AG lançou sua próxima geração de ERPs, conhecidos como R/3. Nos anos seguintes muitas das grandes corporações começaram a investir nos ERPs oferecidos pela SAP, Oracle, Baan, entre outras.

2.3.2 Composição

Tais sistemas computacionais são comumente compostos por módulos pela inerente personalização para os clientes e pelas facilidades na negociação de pacotes empresariais.

Exemplos de módulos típicos incluem Recursos Humanos, Manufatura, Logística, Financeiro, mas não se restringem a estes. Cada módulo deve estar atrelado a processos específicos do negócio, acessar um banco de dados compartilhado ou exclusivo e pode ser considerado uma aplicação única do ponto de vista da interface do usuário [33].

Sua modularidade resulta em restrições e benefícios para o desenvolvimento desta linha de produtos, destacando-se a relativa independência entre as unidades de software que constituem tais módulos, culminando com questões relativas a:

- **Integração dos dados**: Como será o fluxo de dados entre módulos distintos? Onde os dados estarão armazenados e como será seu acesso? E entre sistemas heterogêneos?
- **Gerênciamento de dependências**: Se o Módulo Administrativo, por exemplo, não puder ser acessado (devido a problemas de qualquer ordem), como se dará o funcionamento das suas dependências?
- **Versionamento**: Haverá suporte a diferentes versões de um módulo? Como?
- **Federação**: É suportado o funcionamento como um sistema distribuído? Como?
- **Controle de acesso**: Quais serão os usuários e permissões deste sistema?

Apenas para citar alguns exemplos de questionamentos que delineiam o produto final e caracterizando suas tipicidades, restrições e custos, sempre em alinhamento com os objetivos organizacionais e processos de negócio. Para tanto, define-se formalmente alguns pré-requisitos necessários para um sistema de gestão [37]:

- Sistemas ERP são desenvolvidos a partir de modelos-padrão de processos
- Sistemas ERP são integrados
- Sistemas ERP têm grande abrangência funcional

Figura 2: Alinhamento dos sistemas organizacionais com a organização

- Sistemas ERP utilizam um banco de dados corporativo
- Sistemas ERP requerem procedimentos de ajuste

Que devem ser respondidos pela solução tecnológica, destacando-se, por fim, uma característica intrínseca [37]:

"*Os sistemas ERP realmente integrados são construídos como um único sistema empresarial que atende aos diversos departamentos da empresa, em oposição a um conjunto de sistemas que atendem isoladamente a cada um deles*".

Reforçando a idéia de integração entre informações, abordada mais profundamente adiante (seções 3.3 e 3.2.2).

2.3.3 Implantação nas organizações

Ao invés de apresentar uma lista extensiva de tópicos cobrindo todos os possíveis benefícios e malefícios das soluções de gestão existentes, decidiu-se por destacar dados provenientes de uma pesquisa recente (2010, [24]) realizada em pequenas e médias empresas em escala global e que se presta a balizar o retorno comum as soluções existentes

Destacam-se, nestes, a ínfima parcela dos gerentes de TI entrevistados que não conseguiram distinguir os benefícios de sua implantação, bem como o tempo de implantação das soluções, que comumente excede expectativas de curto prazo.

Foram também selecionadas duas perguntas estratégicas tocante a implantação de sistemas de gestão nas organizações.

A tecnologia traz consigo valor estratégico?

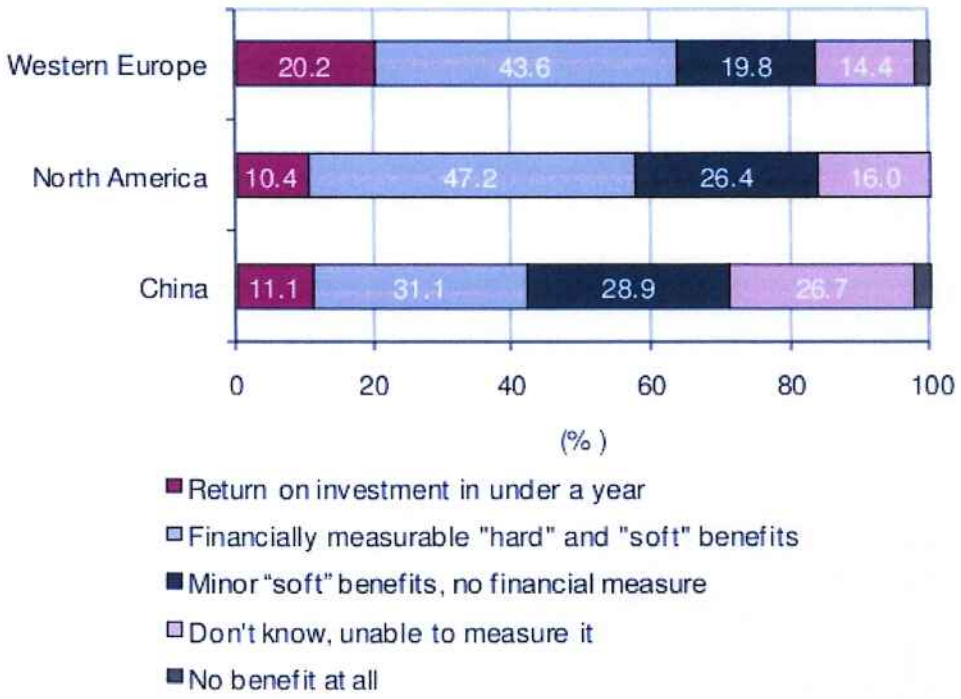

Figura 3: Beneficios recebidos sobre a solução ERP adotada [24]

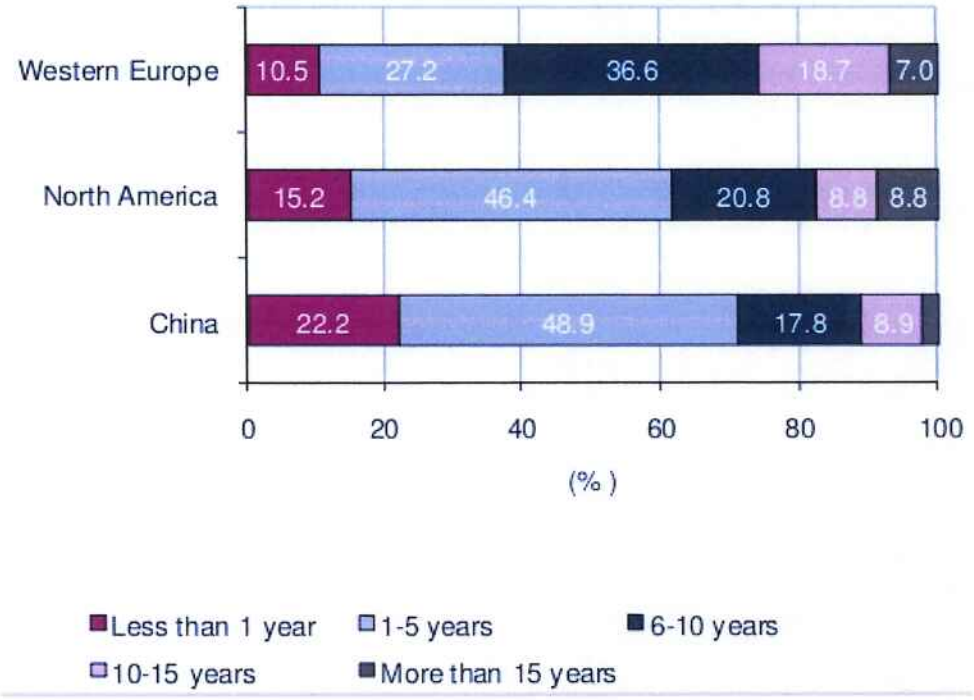

Figura 4: Tempo para a implantação de uma solução ERP [24]

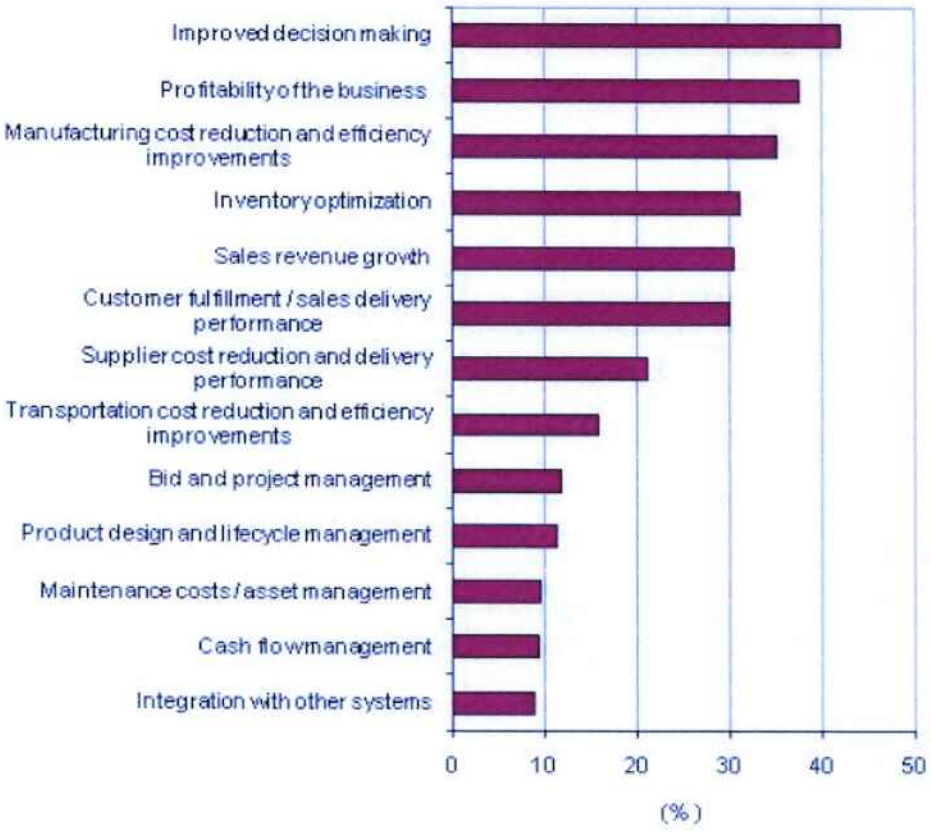

Figura 5: Áreas onde os sistemas ERP proveram contribuições significativas [24]

O poder da tecnologia vem crescendo e se expandindo de tal modo que cada vez mais empresas chegam a vê-la como um recurso cada vez mais crítico para a obtenção de sucesso, fato claramente refletido nos seus hábitos de consumo. Segundo um estudo do U.S. Department of Commerce's Bureau of Economic Analysis, em 1965, menos de 5% dos gastos de capital de organizações americanas eram destinados a área de TI. Após a introdução do computador pessoal no começo da década de 80 esta percentagem subiu para 15%. Ao final da década de 90 tinha-se quase 50%. [9]

Sob a mudança de pensamentos reside uma suposição simples: tal como a força e a ubiquidade da TI aumentaram, o mesmo ocorreu com seu valor estratégico. Suposição lógica, intuitiva, mas errada.

O que torna um recurso realmente estratégico, dando base para uma vantagem competitiva sustentada, é a sua escassez, não sua ubiquidade [9]. Pois só se obtém vantagem competitiva a partir de ações ou artefatos que seus concorrentes não possuem. As principais funções da tecnologia, hoje, tais como o processamento, armazenamento e transporte de dados, se tornaram acessíveis para todos, tornando-se um preço a pagar a fim de realizar negócios. Dito isso, conclui-se que a tecnologia, por si só, não representa vantagem competitiva para as organizações, mas sim o

entendimento do melhor modo de uso da tecnologia - Google ou Dell figuram como exemplos destacáveis.

No caso de sistemas de gestão, tal entendimento é traduzido pela reengenharia dos processos de negócio da empresa após a inserção do elemento tecnológico, a compatibilidade entre tecnologias (por meio da aderência a padrões) ou a flexibilidade das soluções - elementos ainda escassos ou dotados de unicidade diante de soluções milionárias e altamente inflexíveis com alta dependência tecnológica (vide 2.2) e contratos de serviço, não apresentando projeto orientado ao reúso, por exemplo (vide 3.3).

É possível investir a qualquer tempo em TI?

A armadilha na qual muitos constantemente caem é assumir que as oportunidades de vantagem estarão disponíveis indefinidamente. De fato. a janela na qual é possível obter vantagem esta aberta apenas por um curto período de tempo.

Quando o potencial tecnológico começa a ser amplamente apreciado, grandes quantias de dinheiro são inevitavelmente investidas até que este se torne um preço a pagar para não sucumbir frente aos concorrentes.

2.4 Considerações finais

Neste capítulo foram introduzidos os fundamentos básicos para o projeto em questão, a partir dos quais é possível inferir a importância do mesmo, dado o gigantesco mercado existente para pequenas empresas, e do modelo adotado no projeto, o outsourcing por meio de cloud computing, modelo particularmente interessante pois, dentre outros motivos, permite o crescimento da empresa junto aos seus gastos de TI, sem demandar altos investimentos iniciais para tanto.

Dentre os tipos de nuvens existentes destaca-se que o projeto apresenta-se como SaaS, por tornar desnecessarias etapas de intalação no cliente, apresentando-se funcionalmente por uma interface web, e faz uso de nuvens de serviço PaaS, como será explicado adiante.

Destaca-se, por fim, que o projeto se insere no contexto de sistema de gestão empresarial com as características citadas (*Enterprise Resource System*) e apresenta-se composto como apresentado anteriormente.

- **Descoberta**: Serviços são suplementados com metadados por meio dos quais podem ser efetivamente descobertos e interpretados.

- **Composição**: Podem atuar no papel de elementos de agregados maiores independente do tamanho e da complexidade da composição.

Outro papel importante na orientação a serviço é o de **consumidor do serviço**, no qual um componente arquitetural está engajado na transferência de dados com um **provedor de serviço** - papel do serviço, neste momento.

Note, por fim, que a abordagem aqui descrita é adotada visando atingir os objetivos:

- **Aumento da operabilidade intrínseca**: O projeto de serviços para a compatibilidade natural de modo que possam ser efetivamente reconfigurados em resposta a mudanças nos requisitos do negócio.

- **Aumento no nível de federação**: O estabelecimento de uma camada contratual uniforme que abstrai as não uniformidades das camadas subjacentes permite o estabelecimento de conexões em um agregado de nós de maior porte.

- **Aumento na diversificação de opções de fornecedores**: Uma abordagem orientada a serviços, por ser fundamentada em um modelo arquitetural neutro de fornecedores, permite que a organização desenvolva sua arquitetura em alinhamento com o negócio sem se limitar a detalhes tecnológicos atrelados a fornecedores.

- **Aumento no alinhamento entre negócio e tecnologia**: Redução da defasagem tecnológica causado pelo projeto orientado ao negócio em um contexto funcional, permitindo a reflexão do negócio na tecnologia.

- **Aumento no Retorno sobre o Investimento**: A visualização dos serviços como ativos tecnológicos onde se espera que os mesmos forneçam valor repetido, superando o custo da entrega e posse dos mesmos.

- **Aumento na agilidade organizacional**: Requisitos novos e em constante mudança podem ser atingidos mais rapidamente por meio do estabelecimento de um ambiente no qual as soluções possam ser personalizadas com menor esforço, priorizando o reúso e a interoperabilidade nativa de serviços existentes.

Apenas para citar alguns. Há, deste modo, uso real e comprovações diversas, em ambientes de produção, da sua eficácia quando no contexto de um projeto que o implementa corretamente.

3 Disponibilização e consumo de serviços

3.1 Arquitetura de software

A computação vem buscando resolver problemas relacionados a complexidade desde a sua concepção. Os problemas mais recentes nesse sentido vem sendo abordados com o uso de estruturas de dados específicas, algoritmos novos e por meio da divisão do problema em partes menores.

Nesse sentido, a arquitetura de software de um programa ou um sistema de computação é a estruturação de seus componentes, abrangendo seus componentes de software, as propriedades visíveis destes componentes e seus relacionamentos [20].

Roy Fieldings, um dos criadores do protocolo HTTP, em sua dissertação de doutorado [20] possui uma definição coerente [2] para o propósito visado neste documento:

"*A software architecture is an abstraction of the run-time elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture*".

Observa-se, deste modo, que a abstração figura como um elemento primordial, permitindo que se esconda os detalhes de uma tecnologia em particular para melhor identificar e destacar propriedades específicas nas quais deseja-se dar ênfase.

Neste caso, um sistema complexo teria muitos níveis de abstração, cada qual com sua própria arquitetura, que, por sua vez, é constituída pela interconexão de elementos por meio de *interfaces*.

Seus elementos majoritários, segundo o autor, são:

- **Componentes**: Elementos de processamento que realizam transformações nos dados
- **Conectores**: Conectam diferentes partes da arquitetura em questão
- **Dados**: Elementos que contém a informação em questão

Neste caso, um componente é uma unidade abstrata de instruções de software que provê a transformação dos dados por meio da sua interface. Possíveis exemplos incluem a realização de cálculos, a adaptação de conteúdos entre formatos ou o encapsulamento de dados.

Um conector é responsável pelo intermédio da comunicação, coordenação e cooperação entre componentes, podendo consistir em elementos de um subsistema que realiza codificação dos dados, a transferência, e a decodificação dos dados visando a transferência e a entrega segura dos dados.

[2] Bem como notória, tenha em vista seu uso pela W3C [40]

Há, por fim, os dados, que são elementos de informação transferidos entre componentes por meio de conectores. Os primeiros se diferem dos últimos por não possuirem uma finalidade, caracterizando-se por serem apenas representações da informação, tal como um número isolado, que nada representa.

3.2 Orientação a serviço

A orientação a serviço é um paradigma de projeto destinado a construção de unidades lógicas (**serviços**) individualmente projetadas de modo que possam ser utilizadas coletivamente e repetidamente visando atingir metas estratégicas ou benefícios quando associados a Arquiteturas Orientadas a Serviços (SOA, 3.3) [17].

Tal paradigma tem recebido muita atenção nos tempos recentes pelas organizações devido aos benefícios prometidos, tais como maior retorno sobre o investimento ou a interoperabilidade, benefícios que culminam com baixos custos no desenvolvimento ou manutenção de softwares.

3.2.1 Princípios e benefícios

A aplicação de princípios de projeto orientados a serviço faz a distinção entre unidades lógicas que figuram como serviços de outros tipos de unidades lógicas, tais como objetos ou outros tipos de componentes arquiteturais. São eles:

- **Contrato de serviço padronizado**: Serviços dentro de um mesmo inventário de serviços estão em conformidade com as mesmas padrõnizações, a fim de prover uniformidade a um conjunto de serviços.
- **Baixo acoplamento**: Contratos de serviço resultam em baixos requisitos de acoplamento e são em si mesmos desacoplados do ambiente que os cerca.
- **Abstração**: Contratos de serviço contém apenas as informações essenciais e a informação sobre os serviços estão limitadas ao que está publicado nos contratos de serviço.
- **Reúso**: Serviços contém e expressam uma lógica agnóstica e podem ser posicionados como recursos empresariais reutilizáveis, resultando em menores custos de desenvolvimento do produto.
- **Autonomia**: Serviços exercem uma grande parcela de controle sobre o ambiente de execução sobre o qual estão apoiados.
- **Falta de estado**: Minimização de consumo de recursos pela gerência de informações de estado sob demanda. Resulta em menores esforços de escalabilidade e baixos custos.

3.2.2 Tecnologias

O W3C define Web service como "um sistema de software construído para suportar interoperabilidade em uma interação máquina-a-máquina sobre uma rede"[39]

Os Web services, no entanto, são um conceito abstrato e devem ser implementados por um componente de hardware ou software que troquem mensagens entre si. Esses componentes são chamados de agentes. O serviço por sua vez é o conjunto de funcionalidades disponibilizadas e independe do agente. Para prover algum serviço, a entidade provedora deve disponibilizar uma agente que possa implementar esse serviço. Já a entidade requisitora deve ter um agente requisitor que possa trocar mensagens com o agente provedor para requisitar o serviço. Para se utilizar o Web service é necessário ter conhecimento de sua estrutura e os padrões de troca de mensagens envolvidos. Para isso existe uma especificação que pode ser processada por uma máquina, o Web service description (WSD). Nele estão presentes informações sobre o formato da mensagem, os tipos de dados envolvidos, os protocolos de transporte utilizados e os formatos de serialização dos dados na troca de mensagens entre os agentes requisitores e provederes. O WSD é descrito em uma linguagem chamada WSDL e, além de informações sobre a troca de mensagens, ele especifica uma ou mais localizações do agente provedor na rede. O bom uso do Web service parte do pressuposto que tanto a entidade disponibilizadora quanto a requisitora devem ter conhecimento da finalidade dele. O W3C define as características do Web service ao responder mensagens como a sua semântica. A semântica pode estar documentada como o WSD, ser definida informalmente ou estar subentendida entre os agentes. Abaixo segue a arquitetura de comunicação entre os agentes [39].

Geralmente os sistemas interagem com o Web service utilizando o protocolo HTTP e a informação trocada contém dados serializados no formato XML, além de utilizar outros padrões Web. Existem muitas classes e modos de se utilizar Web services em sistemas, mas no escopo de nosso projeto especificamente será utilizada a arquitetura REST (Representational State Transfer).

O REST parte da idéia de troca de mensagem cliente-servidor, onde utilizando protocolo HTTP, somente a partir das mensagens é possível responder a requisição. Ou seja, nenhuma informação adicional precisa ser armazenada no servidor. É uma arquitetura de Web service muito mais simples que SOAP e WSDL [22].

Como o REST é baseado todo em cima do HTTP, é possível utilizar as operações básicas do HTTP (POST, GET, PUT e DELETE). Logo, é possível utilizar um GET para obter a o resultado de um serviço.

Web services REST podem ser invocadas através da URI associada a ele. O que é perfeito para sistemas distribuídos, pois somente é necessário ter o conhecimento do URI do Web service, não sendo necessário saber onde o serviço está localizado. Então numa aplicação cliente pode acessar inúmeros REST services de forma transparente.

Através do REST é possível trabalhar com recursos através de sua representação em formato XML. Logo é possível obter entidades no formato XML de um sistema que representam um objeto de negócio, como um pedido de venda, fazendo uma requisição para a URI do serviço que disponibiliza os dados. Então é necessário para sistemas que utilizam REST ter a capacidade de serializar as entidades que serão trocadas nas requisições dos Web services.

3.3 O paradigma SOA

A arquitetura orientada a serviço (SOA) é um paradigma arquitetural que visa aumentar a agilidade e a eficiência de uma organização, reduzindo o retrabalho no desenvolvimento de software. Mais formalmente:

"*Service Oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations*" - OASIS SOA Reference Model [31]

Para tanto, faz uso dos princípios de Orientação a Serviço, extrapolando-os em alguns pontos. Seus princípios básicos são [17]:

- **Orientado ao negócio**: Alinhamento da tecnologia com o negócio. Princípio, este, de aplicação contínua, a fim de evitar grandes defasagens entre a tecnologia e o negócio. Destaca-se o fato que estas ocorrem gradualmente e resultam em uma diminuição do potencial de preenchimento dos requisitos de negócio, gerando dificuldades para se adaptar à mudança de requisitos constantes de negócio, culminando, por fim, a troca de softwares, que zera a defasagem e reinicia o ciclo.

- **Neutralidade**: O modelo arquitetural não deve se basear em tecnologias proprietárias de um fornecedor, possibilitando a combinação e substituição de elementos do mesmo. O projeto SOA sob tutela de uma tecnologia específica traz consigo elementos desta, inibindo a evolução futura em resposta a inovações que possam se tornar disponíveis de outros fornecedores.

- **Escopo organizacional**: O escopo da arquitetura deve ser aplicado ao nível da organização, permitindo o reúso e a composição dos serviços entre divisões organizacionais e viabilizando a visão dos serviços como ativos empresariais.

- **Centrados na composição**: O projeto arquitetural deve abranger o uso e agregação de serviços, permitindo mudanças constantes de modo rápido e eficiente. Para tanto, o mesmo

deve ser feito tendo em vista cenários de composição simples e complexos. Decisões arquiteturais (e extensões de infra-estrutura correlatas) ligadas a escalabilidade, confiabilidade, processamento de transações e integridade são essenciais para tanto.

Por meio destes, muitas organizações chegaram a atingir seus objetivos organizacionais, como por exemplo:

- **Redução de custo**: A exemplo da Verizon Communications, que clama possuir de 2.5 a 3 milhões de transações web services por dia, fazendo uso de uma arquitetura SOA própria, denominada *IT Workbench* e lançada em 2004. Segundo a mesma, os custos de TI foram reduzidos em 50% pela eliminação de sistemas redundantes e integração da operação dos quase 7000 desenvolvedores [28].

- **Gerência de muita informação**: Tal como feito pela National Aeronautics and Space Administration (NASA) em seu sistema denominado Earth Observing System Clearinghouse (ECHO), lançado em 2002, e que visa permitir o acesso aos petabytes de dados de cada nave espacial, integrando os, outrora heterogêneos, sistemas distribuídos [27].

- **Melhora de serviços para o cliente**: A fim de permitir serviços online para clientes, a Harvard Medical School construiu uma arquitetura orientada a serviços, integrando cerca de 400 departamentos por meio de 25 categorias de web services e reduzindo custos em cerca de $1 milhão por ano [29].

3.4 Futuro do SOA

Em 2004, o termo *Web 2.0* foi cunhado em uma conferência da O'Reilly Media, referindo-se a uma assim chamada "*segunda geração"de aplicações web, caracterizadas por um grau maior de interação e colaboração entre usuários*". Desde então o termo passou a ser constantemente utilizado pelo mercado, na esteira do rápido crescimento de um número significativo de blogs, comunidades virtuais, wikis e outras aplicações.

A web, na verdade, não havia mudado, como não mudou até hoje. O que mudou foi a capacidade de transmissão de dados entre servidores e até o cliente, dando margem a novos tipos de conteúdo sobre as mesmas padronizações das décadas de 70-80, que refletiu em mudanças nas formas de uso das mesmas. Foram, desta forma, cunhados termos novos para, essencialmente, as mesmas tecnologias, sujeitas a uma evolução *constante* em seu *uso*, tais como AJAX ou Web 2.0.

Segundo O'Reilly [25], dentre as principais características inerentes à filosofia Web 2.0, destacam-se:

- Uma plataforma para a construção de sistemas interligadas por um conjunto de protocolos, padrões abertos e acordos de cooperação
- O maior uso da inteligência coletiva advinda dos usuários da web
- O uso de mídias e dispositivos diversos para o consumo de aplicações baseadas na internet

Em outras definições são destacadas suas características de "*plataforma ou ferramenta que auxilia os usuários na criação e compartilhamento de conteúdo com uma ampla audiência*", tocante a agregação (*mashup*) de conteúdo, destacando o fato de que "*as plataformas de agregação permitem que usuários obtenham conteúdo ou funcionalidades de fontes arbitrárias, combinando-as e expondo-as para o reúso por outras aplicações*" [25].

Note as semelhanças com o paradigma SOA, apresentado anteriormente (seção 3.3), nas suas noções de reúso e composição de serviços, sua afinidade para colaboração e baixo acoplamento entre serviços possívelmente distântes e heterogêneos, e, por fim, o princípio compartilhado de agilidade e apoio a mudanças permanentes de caráter estrutural.

Há, porém, divergências destacáveis entre ambos, como por exemplo o caráter social das aplicações Web 2.0, no sentido que estas facilitam a interação humana e lidam com conteúdo legíveis por seres humanos e, comumente, de caráter mais simples, refletido pelas tecnologias subjacentes, tais como o REST ou o RSS. O SOA, entretanto, se destina a comunicação otimizada, entre máquinas, comumente binária, não legível ou de caráter mais complexo, representada por sua vez pelas tecnologias SOAP, WSDL e Fast Web Services. Convem também citar que há, na web, uma maior necessidade de apresentação e integração com interfaces, menos comum em cenários de uso do SOA. [25]

Nesse sentido, diante da crescente importância dos serviços no contexto empresarial, e dos padrões web, no mundo inteiro, houve, naturalmente, uma evolução na nomenclatura utilizada para caracterizar os novos usos das tecnologias, e, mesmo, as novas tecnologias, de modo que falar em Web 2.0 tornou-se passado. Há, na verdade, muitos termos distintos para caracterizar o momento no qual vivemos, dentre eles *Internet of Services*, Web 3.0 e *Web of Things*, dentre os mais recentes, em uma alusão à crescente importância que a web vem ganhando, suplantando sistemas operacionais e serviços individuais.

3.4.1 Internet of Services

Até mesmo os céticos admitem a crescente facilidade tocante à integração entre serviços [21], divergindo sobre o futuro dos mesmos. Afirmam que nunca será facil integrar serviços distintos devido a incompatibilidades entre estruturas de dados e processos de negócios muitas vezes distintos.

Há, por outro lado, trabalhos que sugerem rumos mais promissores para o futuro da web e o papel dos serviços na mesma, criando, dentre outros, o conceito de *Internet of Services* [25], fusão de ambos, visando trazer aos usuários comuns o uso e gerência de serviços que, outrora, estavam restritos ao uso profissional dentro do contexto empresarial.

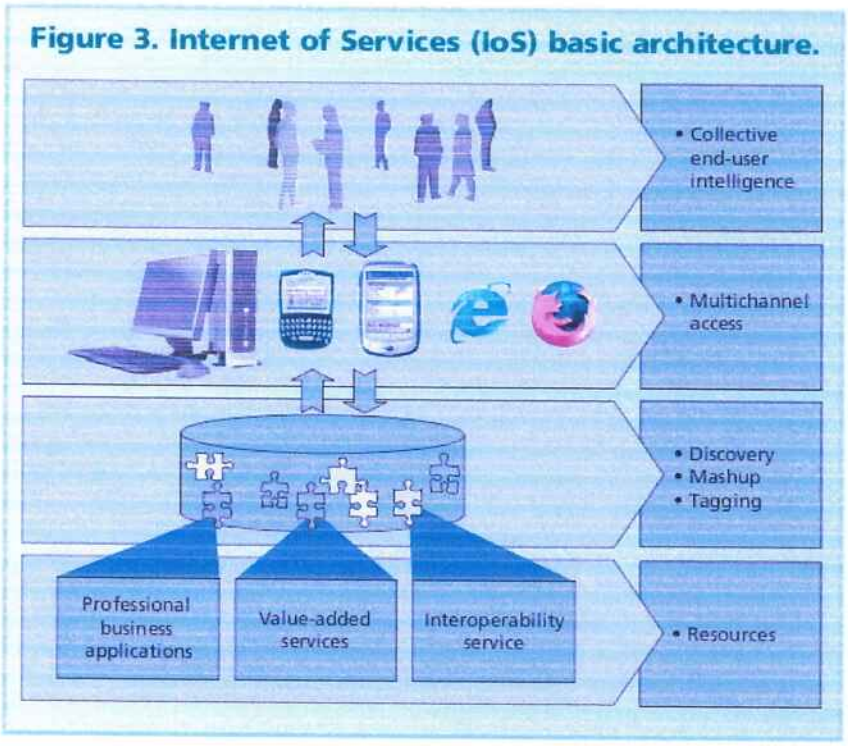

Figura 6: Arquitetura básica Internet of Services [25]

Neste cenário, os recursos são acessíveis pela web e são registrados de modo que possam ser facilmente descobertos, "etiquetados" (*tagging*), a fim de coletar feedbacks dos usuários a respeito dos mesmos, e sujeitos a *mashup*, composição e interligação dos mesmos visando a criação de novos serviços.

Destaca-se o caráter descentralizado deste mercado, no qual os usuários acessam estas plataformas por meio de mecanismos de descoberta utilizando interfaces intuitivas que facilitam as atividades de mashup e tagging, sem fazer uso de meios técnicos para tal, como linguagens de programação. Note a abstração da tecnologia subjacente, possibilitando, cada vez mais, que a composição dos serviços seja cada vez mais uma atividade imaginativa, em detrimento as atividades técnicas, supridas pelas padronizações, bem como aos canais de comunicação, tais como celulares ou computadores, cada vez mais em conformidade com padrões web. Em última instância, tais atividades seriam realizadas por um especialista nos processos negócio sem grandes conhecimentos técnicos, por parte da organização, e seriam acessíveis por quaisquer meios com

acesso a internet. São estes os benefícios de arquiteturas abertas e que abrangem possibilidades de seres humanos lidarem com recursos web, gerando novos recursos com visibilidade global.

Os mashups corporativos representam um caso de uso específico deste tipo de arquitetura, onde a combinação de princípios tanto da Web 2.0 (modelo self-service de serviços e inteligência coletiva agregada) e o SOA (composição de blocos de construção reutilizaveis) estão presentes, hoje, em empresas como Amazon e eBay, que "*fizeram um trabalho brilhante com serviços web, abrindo suas infra-estruturas de TI e deixando com que centenas de pequenos vendedores se plugassem nos mesmos*" [21], e podendo ser diferenciais para a aproximação com clientes ou fornecedores.

3.5 Trabalhos relacionados

Foi feito um extensivo estudo de softwares ERP, em especial os de código aberto, a fim de melhor compreender suas entranhas e, portanto, projetar o sistema em questão de forma mais coerente, de "pés no chão". Um trabalho em específico nos auxíliou neste processo, CORREA2008 [12], onde o autor faz um comparativo entre as arquiteturas de software livre existentes.

Para realizar uma análise do estado da arte, é necessário estabelecer um contexto para que os sistemas a serem analisados sejam o mais semelhantes possível com o sistema que estamos propondo. Então, os sistemas que terão mais foco em nossos estudos são aqueles voltados para processos de negócio mais simples que atendam pequenas empresas. Outra característica deve ser a utilização de uma arquitetura de software livre.

Para auxiliar em nossa análise foi utilizado um trabalho que efetua a comparação entre alguns sistemas que atendem as limitações acima mencionadas. Abaixo seguem os principais sistemas:

3.5.1 Adempiere

É um ERP de código aberto e distribuído sobre a licença GPL. Ele é mantido por algumas empresas e seu código está disponível para download. Ele é derivado de outro projeto de ERP, também de código aberto, chamado "Compiere".

A empresa que deseja utilizá-lo deve ter a própria infra-estrutura para isso.

Então, apesar de ser um software livre, podem existir gastos com equipamento e mão-de-obra, o que não seria necessário em nossa proposta. Dentre as suas funcionalidades ele apresenta: controle de estoque, controle de usuários, controle de pagamentos, pedidos e relatórios de vendas, controle de compras, gestão de políticas de marketing. Como se pode ver esse sistema apresenta mais funcionalidades que a nossa proposta e elas são demasiadas complexas para nossos fins.

Foi desenvolvido em Java e tem suporte aos bancos de dados PostgreSQL, Oracle, OracleXE. Sua arquitetura é muito complexa e não atende alguns requisitos de nossa proposta, pois é pouco escalável, o sistema todo fica concentrado em um servidor e para cada versão lançada deve ser

instalado novamente. Além disso, como foi dito anteriormente, o cliente fica refém da infra-estrutura que é utilizada para o sistema.

Apesar das grandes discrepâncias com o que foi proposto em para nosso ERP algumas de suas funcionalidades foram estudadas para ajudar na implementação das funcionalidades semelhantes que estarão em nosso projeto.

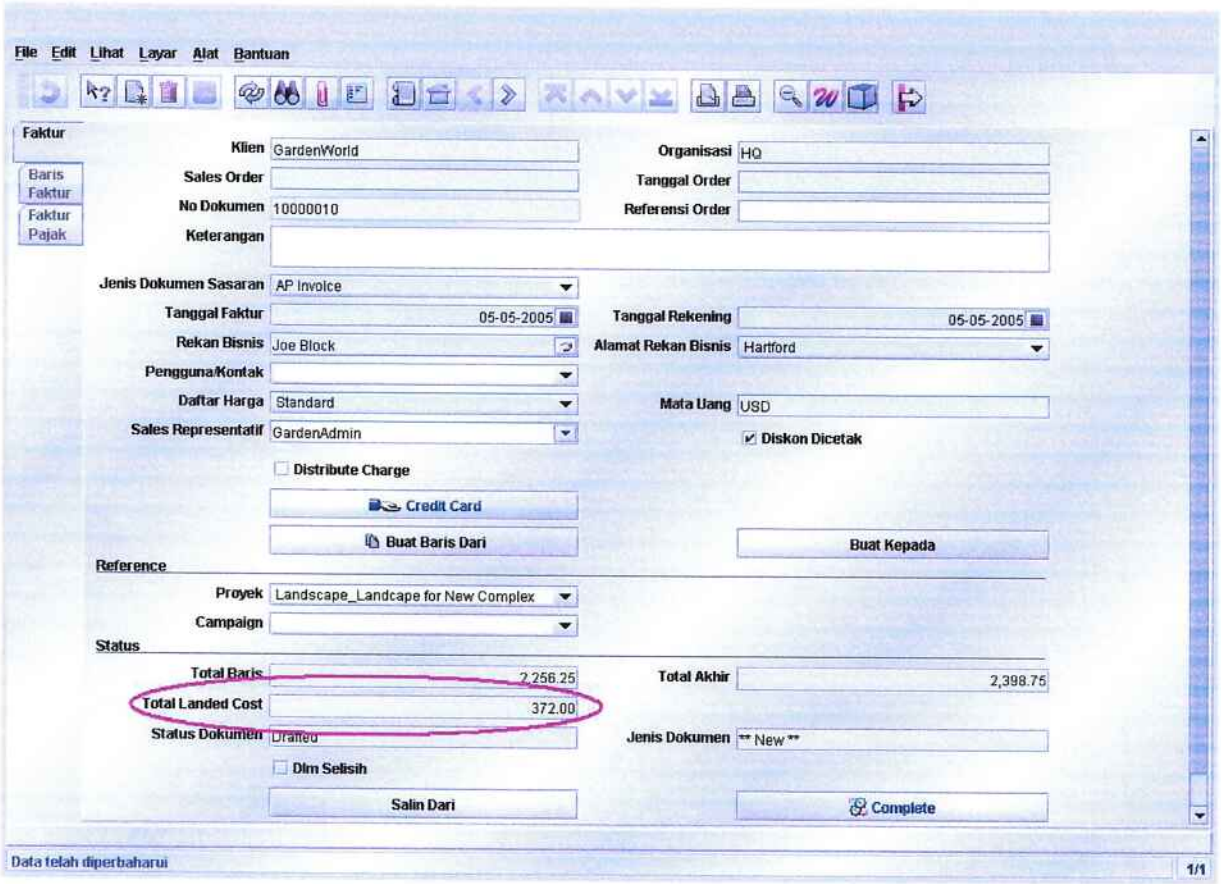

Figura 7: Screenshot de uma tela do Adempiere [19]

3.5.2 Apache OfBiz

Desenvolvido pela "Apache Software Foundation"também é um ERP open source que automatiza muitos dos processos de negócio de empresas. Ele foi implementado na forma de framework, para que pudesse ser expandido ou customizado mais facilmente. Sua arquitetura é dividida em camadas, sendo que elas são: camada de apresentação (Presentation Layer), camada de negócio (Business Layer) e camada de acesso aos dados (Data Layer). Ele utiliza várias bibliotecas open source para o seu desenvolvimento.

Além de fornecer o serviço ERP, o Apache OfBiz ainda oferece funcionalidades como CRM, eCommerce, SCM, MRP. Isso mostra que ele é muito mais complexo que nosso escopo, mas foi útil para ajudar a definir a arquitetura de nosso projeto. Por exemplo, ele utiliza Web service SOAP para implentar os serviços disponíveis na camada de negócios.

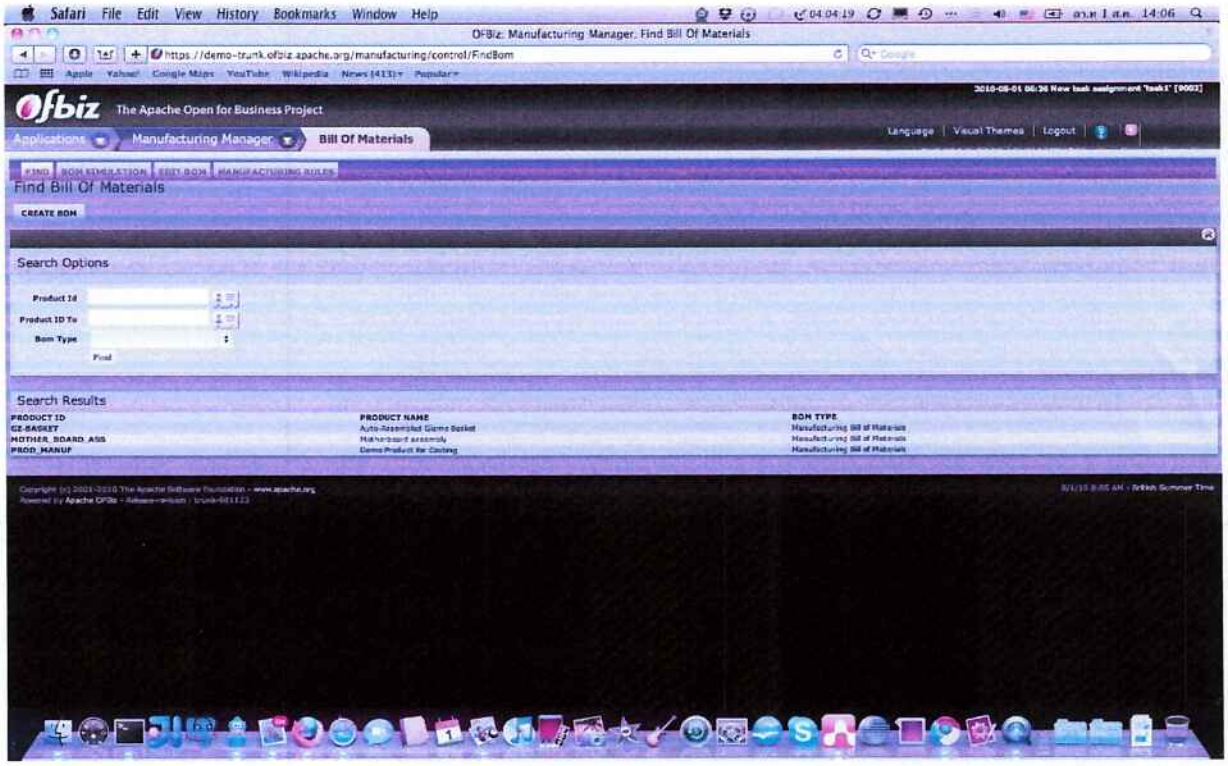

Figura 8: Screenshot de uma tela do Apache OfBiz [8]

3.5.3 webERP

É um ERP open source desenvolvido totalmente para web. Seu código fonte está disponível para download e pode ser instalado em qualquer tipo de servidor, como o Windows IIS e o Apache. Apresenta várias opções de customização como escolha de temas, seleção de idiomas, utilização de criptografia SSL nas páginas. [5]

Ele foi criado para utilizar somente interface HTML e scripts PHP. Utiliza o mínimo possível de scripts Java (JSP) para ser compatível com a maioria dos Web browsers, também sendo possível o acesso tanto de desktops quanto de dispositivos móveis como celular. Apesar de algumas diferenças quanto à proposta de nosso projeto, pois é necessário instalá-lo em um servidor, a proposta de utilizar interfaces HTML e tentar obter maior compatibilidade com browsers e dispositivos reflete o intuito de nosso projeto.

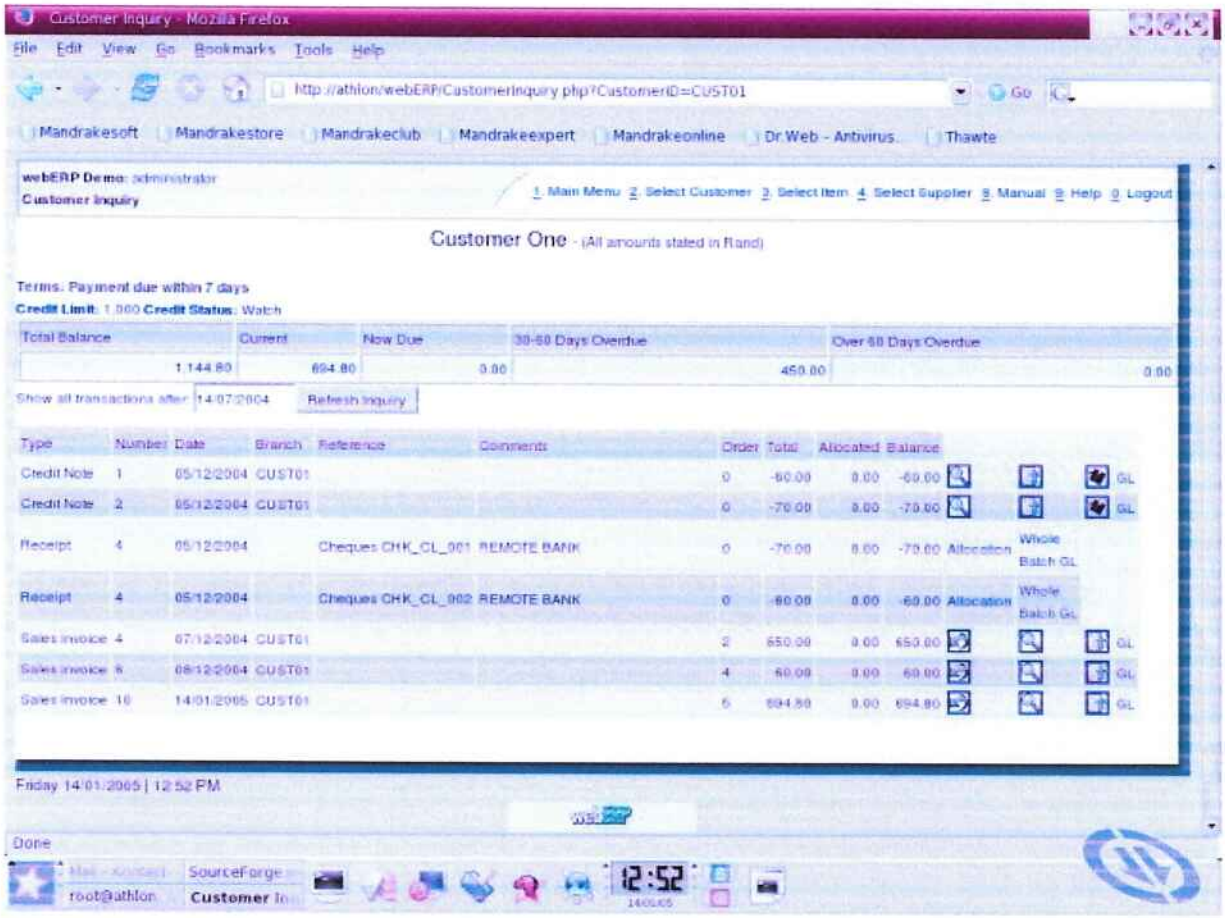

Figura 9: Screenshot de uma tela do webERP [5]

3.5.4 Openbravo

Também é um ERP open source desenvolvido para web que pode ser instalado em servidores tanto quanto em clusters na nuvem. Oferece muitas funcionalidades, sendo que a maioria foge do escopo de nosso projeto, como ferramentas de BI e (Business Inteligence) e de auditoria. Porém como o nosso foi desenvolvido para web e pode rodar na nuvem, mesmo não sendo esse seu propósito exclusivo. A interface com usuário é bem complexa, cheia de detalhes e totalmente customizavel. A arquitetura foi desenvolvida com Java e segundo seus criadores é totalmente escalável.

3.6 Considerações finais

No presente capítulo foram apresentados os fundamentos da orientação a serviço, elemento arquitetural básico do projeto em questão, suprindo os fundamentos necessários para o entendimento da construção arquitetural da solução proposta e justificando sua maior flexibilidade, em

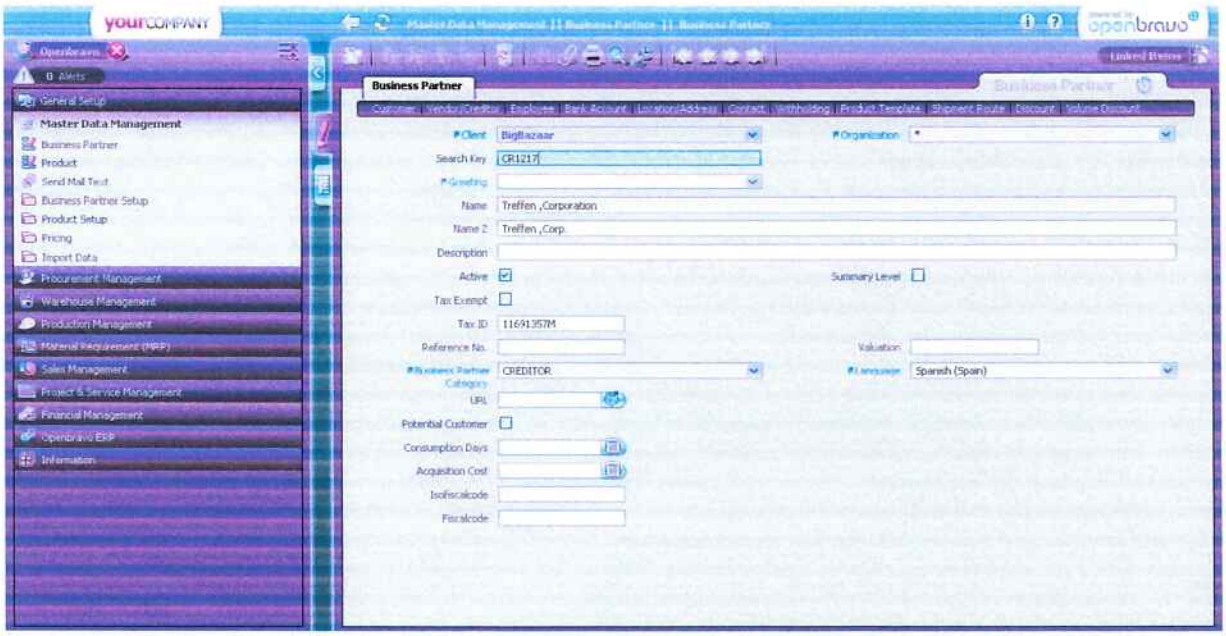

Figura 10: Screenshot de uma tela do Openbravo [3]

termos de mudanças organizacionais, bem como o uso de padrões web visando desacoplar o objetivo visado de fornecedores de tecnologias ou mesmo um conjunto de padronizações com baixa aceitação. Este capítulo findou-se com um estudo dos projetos correlatos existentes no mercado e suas diferentes abordagens na solução dos mesmos problemas.

4 Projeto-proposta

"Simplicity is prerequisite for reliability."

Edsger W. Dijkstra

4.1 Contexto inicial

No presente trabalho é proposta uma arquitetura flexível orientada a serviços (SOA) de sistemas de gestão empresarial (ERP) cujo projeto foi pensado já visando sua utilização em arquiteturas de cloud computing. Destaca-se, também, que esta solução foi pensada como sendo de pequeno porte, visando micro e pequenas empresas, e, portanto, deve contemplar baixos custos de infraestrutura.

4.2 Metodologia de desenvolvimento

A princípio o projeto teve 5 grandes fases, que são:

1. Estudo de tecnologias associadas
2. Proposta
3. Análise
4. Desenvolvimento
5. Testes

A fase de estudo de tecnologias permitiu ter o embasamento técnico para efetuar a proposta, análise, desenvolvimento e os testes do projeto.

Foi efetuado um estudo aprofundado, principalmente, sobre os temas que tem maior impacto no projeto, que são:

1. Sistemas organizacionais
2. Sistemas ERP
3. SOA
4. BPM
5. Arquiteturas de software
6. Cloud Computing

Cada tópico acima contribui de alguma forma para a construção do projeto, desde a fase de análise (como é o caso do item 4, que nos forneceu as ferramentas para a modelagem dos processos de negócio) até a fase de desenvolvimento (estudo sobre SOA e sistemas ERP) e implantação (infra-estrutura em "Cloud Computing", item 6).

Na segunda fase do projeto foi realizada uma proposta do projeto, que definia o escopo e os processos de negócio iniciais (é importante ressaltar que os processos de negócio mudaram durante o decorrer do projeto, de forma iterativa).

A terceira e quarta fase ocorreram de forma iterativa, ou seja, a princípio foi realizado uma análise do sistema a ser desenvolvimento, que teve como resultado os seguintes artefatos:

1. Modelagem dos processos de negócio (análise de processo de negócio)
2. Documento de casos de uso (engenharia de software)
3. Composição dos serviços SOA (análise SOA)
4. Diagrama de classes das entidades e das camadas do sistema (engenharia de software)
5. MER do banco de dados (engenharia de software)
6. Composição básica da arquitetura genérica do sistema (análise de arquiteturas genéricas e de outros sistemas ERP)

O desenvolvimento do protótipo nos permitiu enxergar melhoras na fase anterior, de forma que uma segunda análise foi feita, impactando novamente no desenvolvimento (esse processo ocorreu sucessivamente, até concluirmos que a análise estava, de fato, feita de forma totalmente correta).

A quinta fase também permite que uma nova análise seja realizada em cima do sistema, impactando novamente no desenvolvimento do sistema.

Em resumo, podemos definir que o desenvolvimento ocorreu através de um conjunto de atividades e processos de forma iterativa e sistemática, definida no modelo abaixo:

Observação: detalhes sobre a análise SOA no capítulo 3.

A fase de análise também possui uma metodologia própria, que constitui basicamente de:

1. Análise dos processos de negócio para identificar as principais regras do sistema, partes a serem automatizadas e funcionalidades importantes (análise de negócio)
2. Composição dos serviços para atenderem as funcionalidades definidas no item anterior (análise SOA)
3. Modelagem do diagrama de classe, de forma que permita enxergar as camadas do sistema e o relacionamento entre as principais classes de entidade do sistema

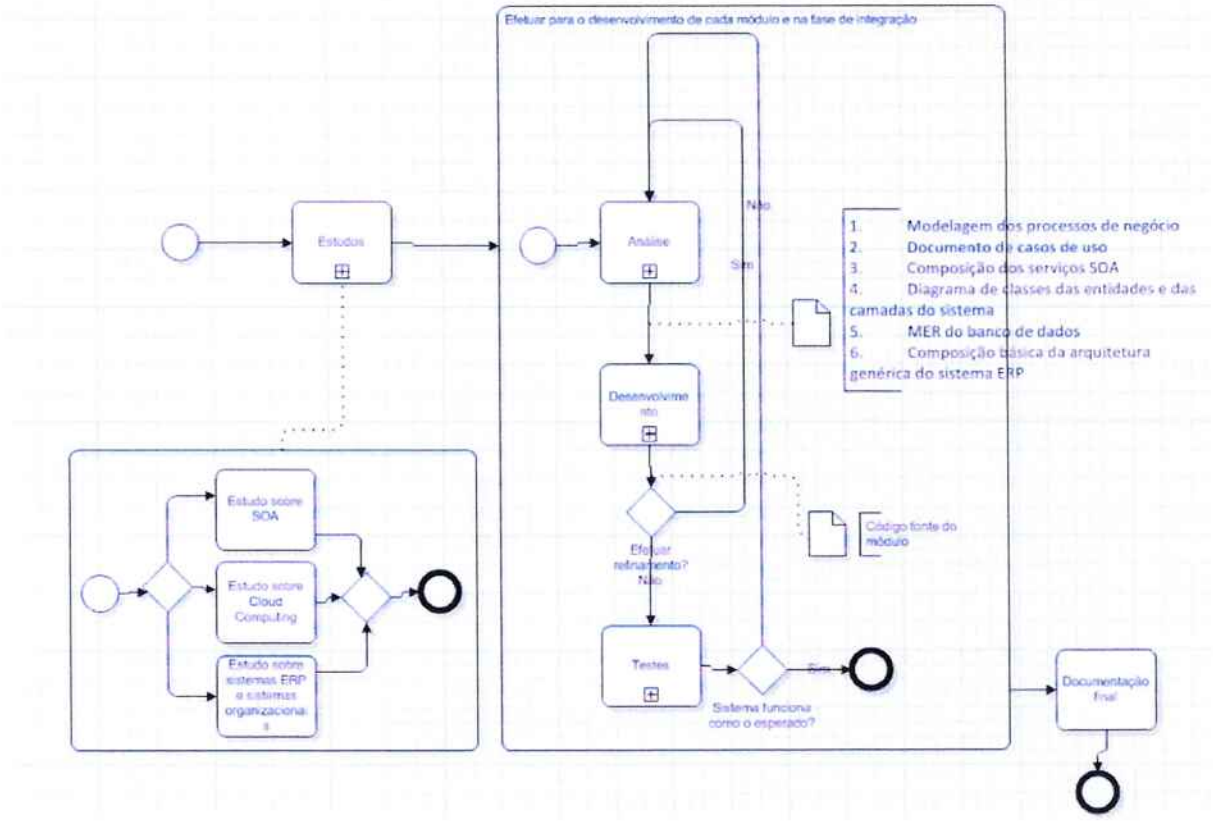

Figura 11: Metodologia de desenvolvimento

4. Modelagem do banco de dados

5. Modelagem da arquitetura entre módulos do sistema, levando em consideração que a estrutura principal do sistema deve funcionar em uma estrutura de nuvem (Cloud Computing)

Cada etapa do processo acima é constituído de um conjunto de atividades a serem realizadas, para termos um controle maior sobre o andamento das atividades e o progresso de cada membro da equipe nas atividades as quais lhe foram designadas, utilizamos a metodologia SCRUM.

4.2.1 Metodologia SCRUM

O Scrum é uma metodologia ágil para gestão e planejamento de projetos de software.

A primeira etapa da metodologia consiste em identificar uma série de atividades a serem realizadas e um conjunto de funcionalidades a serem implementadas, o conjunto de funcionalidades a serem implementadas chama-se Product Backlog.

Os membros da equipe devem definir uma prioridade para cada funcionalidades a ser implementada (ou seja, definir as partes do Product Backlog que são prioritárias).

A segunda etapa da metodologia consiste em dividir esse Product Backlog em conjuntos de funcionalidades a serem implementadas, cada conjunto de atividades a ser efetuado em um determinado intervalo de tempo (pode variar de 2 a 4 semanas) é denominado Sprint.

Os Sprints permitem ver o andamento da equipe dentro de cada etapa do projeto (responde a perguntas como: Esse Sprint será realizado no tempo correto? Caso não seja, qual será o impacto disso no projeto como um todo?).

Ao final da cada Sprint, a equipe testa e apresenta as funcionalidades implementadas.

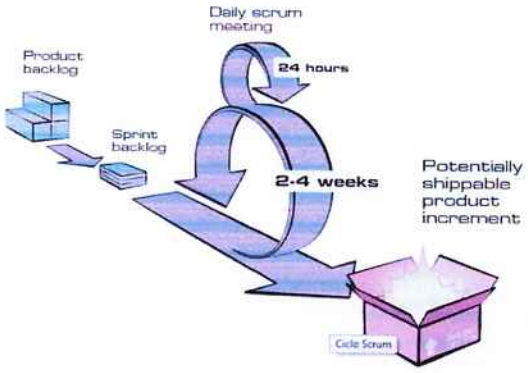

Figura 12: Ciclo Scrum

4.3 Solução proposta

Em termos arquiteturais é muito desejável a aproximação dos modelos de computação distribuída que não fazem uso de compartilhamento de dados, por apresentarem, na maioria das vezes, melhores relações de custo-benefício [35], provendo independência e auto-suficiência a cada nó e evitando, deste modo, pontos de falha e gargalos - característica especialmente interessante para sistemas com requisitos de alta disponibilidade ou escalabilidade.

Na prática, entretanto, é muito difícil obter a aproximação necessária de tais modelos em sistemas que requerem alta integração entre módulos, como, por exemplo, a maioria dos sistemas empresariais, resultando em aplicações distribuídas cujos nós são inerentemente interdependentes. Neste contexto as tecnologias utilizadas figuram como elementos chave pois determinam o nível de acoplamento entre nós pertencentes a uma mesma aplicação distribuída.

Comumente tal tecnologia é regida por padronizações, não apenas pela necessidade de duas tecnologias distintas conversarem na mesma "língua", mas é desejável ter indepêndencia de fornecedores, facilitando o desacoplamento na prática. Nesse contexto, as padronizações web se destacam pela sua notoriedade e grande diversidade de fornecedores que as implementam.

Este trabalho propõe o uso de padrões web a fim de solucionar problemas de acoplamento entre módulos, reduzir *overhead* na transmissão de dados, prover maior flexibilidade arquitetural e, portanto, podendo operar sob um modelo *sob demanda* em nuvens de serviço, culminando, por fim, com redução de custos para a empresa.

4.3.1 Funcionalidades

Essa seção visa explicar o sistema de um ponto de vista funcional (o processo de negócio geral encontra-se na seção 6, anexos, e serve como auxílio para identificar as principais funcionalidades do sistema), é explicado somente o fluxo principal (abstraindo detalhes ou funcionalidades de menor relevância).

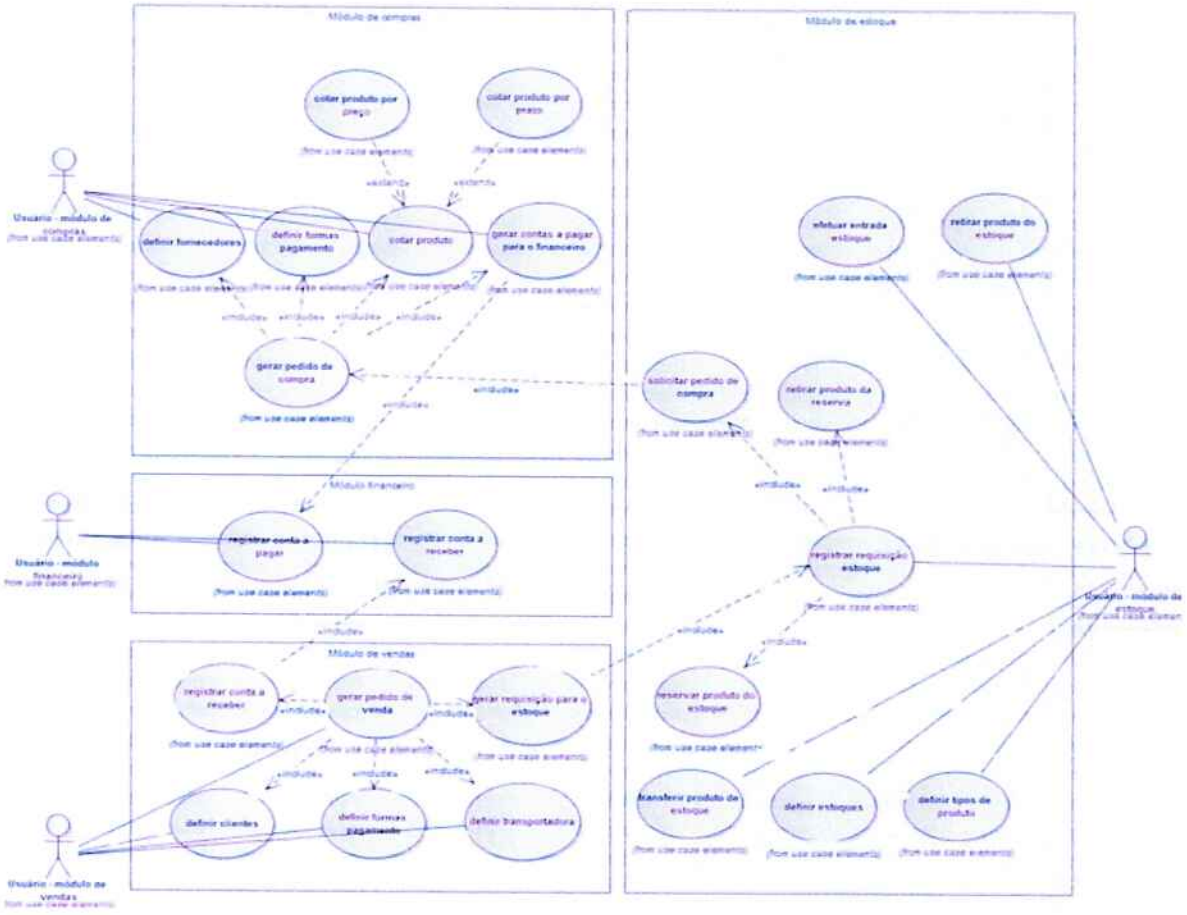

Figura 13: Diagrama de casos de uso do sistema

Vendas O fluxo do sistema começa com o módulo de vendas, após o usuário ter parametrizados possíveis clientes, formas de pagamento e transportadores, ele pode gerar um pedido de venda.

O usuário abre um pedido de venda, escolhe um determinado cliente registrado, define uma forma de pagamento e uma transportadora.

Feito isso, é necessário gerar as duplicatas para o módulo financeiro ("contas a receber"), isso faz com que uma nova opção apareça ("gerar requisições para o estoque").

Cada venda dispara requisições para o estoque (por tipo de produto) para atenderem a uma determinada demanda de um cliente.

Estoque Para fechar uma requisição (atender a demanda solicitada pelo módulo de vendas), o usuário pode reservar produtos que estão no estoque, tirar produtos reservados (para atenderem a demanda de outra requisição, caso seja necessário), efetuar entradas avulsas no estoque (e posteriormente "atrelar" os produtos que "entraram" para a requisição) ou solicitar um pedido de compra.

Compras Pedidos de compra são gerados para atender à demanda solicitada por requisições de estoque, uma requisição pode disparar diferentes compras para atender tal demanda.

Após ter os pedidos de compra gerados pelo módulo de estoque, o usuário tem a opção de definir fornecedores para cada compra gerada (através de serviços de cotação baseados em preços e prazos) e posteriormente agrupá-los por fornecedor.

Com um pedido de compra geral (vários de pedidos de compra agrupados) é necessário definir uma forma de pagamento, gerar as contas a pagar e efetuar a baixa na compra (quando finalizada), que irá efetuar a entrada dos produtos solicitados na compra para o estoque.

4.3.2 Arquitetura

Visando reduzir o acoplamento e prover serviços, como uma solução voltada para pequenas empresas, propomos, neste trabalho, a *utilização de tecnologias web* no desenvolvimento dos processos de negócios. Uma solução, a princípio, simples, mas que apresentou desafios em sua implementação, devido, principalmente a padronizações que não se apresentavam preparadas para funcionamento distribuído, ou mesmo a decisões tocante ao empacotamento modular.

A princípio, cada módulo é implementado dentro de um pacote a nível físico, no nosso caso no formato WAR, amplamente suportado dentre os diversos servidores de aplicação opensource e com suporte ao envio, ativação e desativação de módulos em tempo de execução e remotamente, com alta segurança por parte do administrador e com telas próprias e já inclusas pela infraestrutura. Note que a escolha pela padronização foi adotada sempre quando possível, garantindo os benefícios e a interoperabilidade da mesma, depois, é claro, de outras tentativas que não se apresentaram tão benéficas.

Destaca-se que o formato de empacotamento nesse sistema é irrelevante do ponto de vista da interoperabilidade, uma vez que, ao adotar o modelo RESTful a arquitetura herda as características da web, provendo suporte às diversas linguagens, plataformas e formatos de empacotamento, já que apresentam interfaces uniformes. É, de fato, possível desenvolver uma solução apenas com o conhecimento da interface que a mesma disponibilizará, se o fizer por meio de padrões web, antes mesmo de conhecer mais detalhes sobre seu funcionamento.

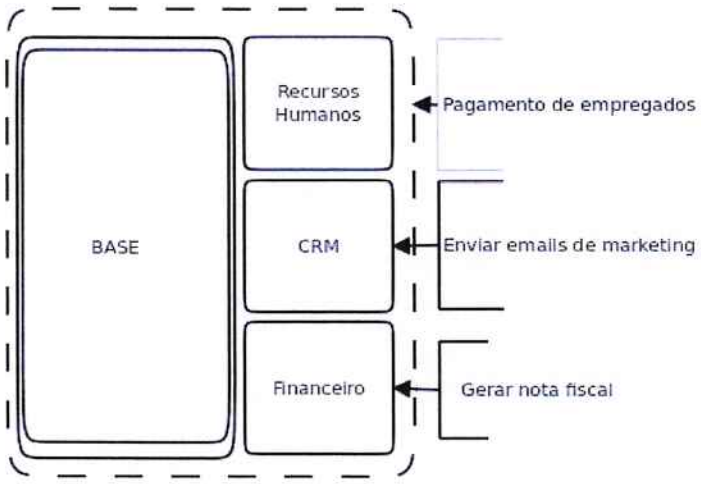

Figura 14: Arquitetura modular projetada na especificação

Tais módulos são conectáveis, no sentido que se enquadram com a arquitetura orientada a serviços (3.3), e são apresentados como seções do componente visual básico (core) - no caso, para cada módulo novo é criada uma aba pré-definida pelas suas configurações na tela principal. Tais módulos disponibilizam serviços seguros seguindo o *estilo arquitetural RESTful*, visando a interoperabilidade entre inúmeros dispositivos e o desacoplamento total das tecnologias do servidor, por meio da utilização de *HTML estático*, evitando, deste modo, acoplamento com tecnologias do servidor - Java Server Pages, dentre outras opções de páginas dinâmicas. Desacoplamento total da camada de visualização, chegando ao nível de se tornar um desacoplamento físico - é executado pelo cliente no consumo de serviços

Outros benefícios do uso de HTML estático incluem a *redução de custos, deixando para o cliente o processamento da camada de visualização e o consumo dos serviços disponibilizados*, acarretando menos processamento nos servidores e por fazer um uso melhor do cache da web, não necessitando requisitar várias vezes as mesmas páginas, podendo ser ainda otimizado, de acordo com a serialização utilizada. Note que a web não foi projetada para os cenários de uso dinâmico de hoje, e em cenários de outsourcing e nuvens de serviço, onde os recursos de rede e processamento são pagos, tais escolhas representam custos muito mais baixos de manutenção.

O desacoplamento de tecnologias proprietárias e o consumo de serviços fazendo uso de padrões web implica no uso de Javascript e AJAX para tanto. Deste modo, o consumo de serviços ocorre do lado do cliente, provendo os benefícios já ditos anteriormente (redução de custos de processamento e rede, desacoplamento das tecnologias proprietárias do lado do servidor), dentre outros característicos a web, tais como a escalabilidade e a redundância[3]. Notou-se, entretanto, que é necessário, mais do que em outros contextos, resolver alguns tipos de problemas novos, tais como:

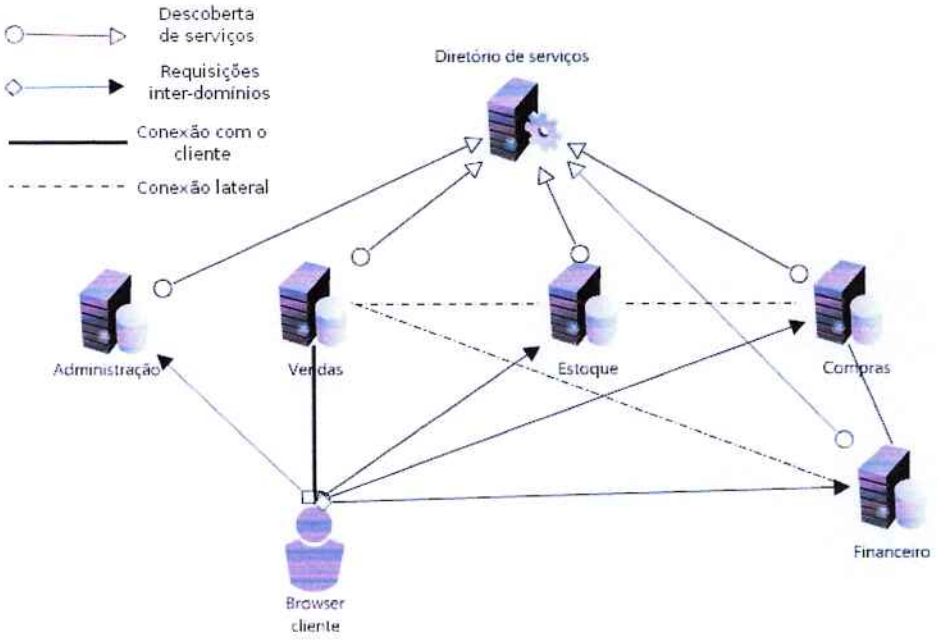

Figura 15: Visão geral da arquitetura de infra-estrutura utilizada

- **Middeware comum**: É necessário desenvolver um middleware comum e robusto para o consumo dos serviços de modo reutilizavel por parte do cliente (consumo dos serviços, manipulação de estruturas de dados, componentes gráficos e gerenciamento de dependências) e do servidor (disponibilização e consumo de serviços).

- **Arquitetura federada**: Quando se consome serviços RESTful a partir do cliente por meio de padrões web há problemas em requisições entre servidores, uma vez que o cliente comumente não possui acesso a serviços disponibilizados por servidores distintos dos quais obteve os executáveis Javascript, pois o browser não o permite (conceito chamado de *sandbox*, que visa prover segurança e evitar ataques XSS). Tal problema foi solucionado por meio

[3] Tenha em vista que a própria internet foi projetada como uma arma militar visando a redundância e a interoperabilidade

de padronizações recentes (outubro, 2010) permitem consumo federado de serviços, dando margem a aplicações web distribuídas a partir do cliente.

- **Diretório de serviços**: A tradução de nomes de serviços para endereços IP para a devida localização dos serviços deve ser feita adequadamente, em uma última instância, por um protocolo Domain Name System, para a maior adequação a padrões. No contexto desse projeto foi criado o papel de Diretório de Serviços, responsável por tal tradução e peça chave no baixo acoplamento entre módulos, pois elimina a necessidade de trechos de código *hardcoded* com o endereço IP de servidores. O diretório de serviços, entretanto, cria a necessidade de registro de módulos no mesmo, quando estes forem ativados, ou a eliminação, caso contrário. Frisa-se, por fim, que este é o único componente que possui estado, problemas comuns para um servidor de nomes, necessitando de um registro seguro de módulos.

Figura 16: Exemplo de tela do sistema final

4.3.3 Arquitetura interna dos módulos

Além da arquitetura entre módulos, o sistema proposto possui uma arquitetura interna própria para disponibilizar os serviços necessários para o aplicativo cliente. Essa arquitetura é composta basicamente de quatro camadas e um framework.

- **Camada de disponibilização dos serviços** : Utiliza servlets 3.0 para disponibilizar para o aplicativo cliente serviços da camada de negócio. aServlet é uma classe Java que apresenta uma interface que possibilita ao pprogramador realizar requisições e obter respostas, em

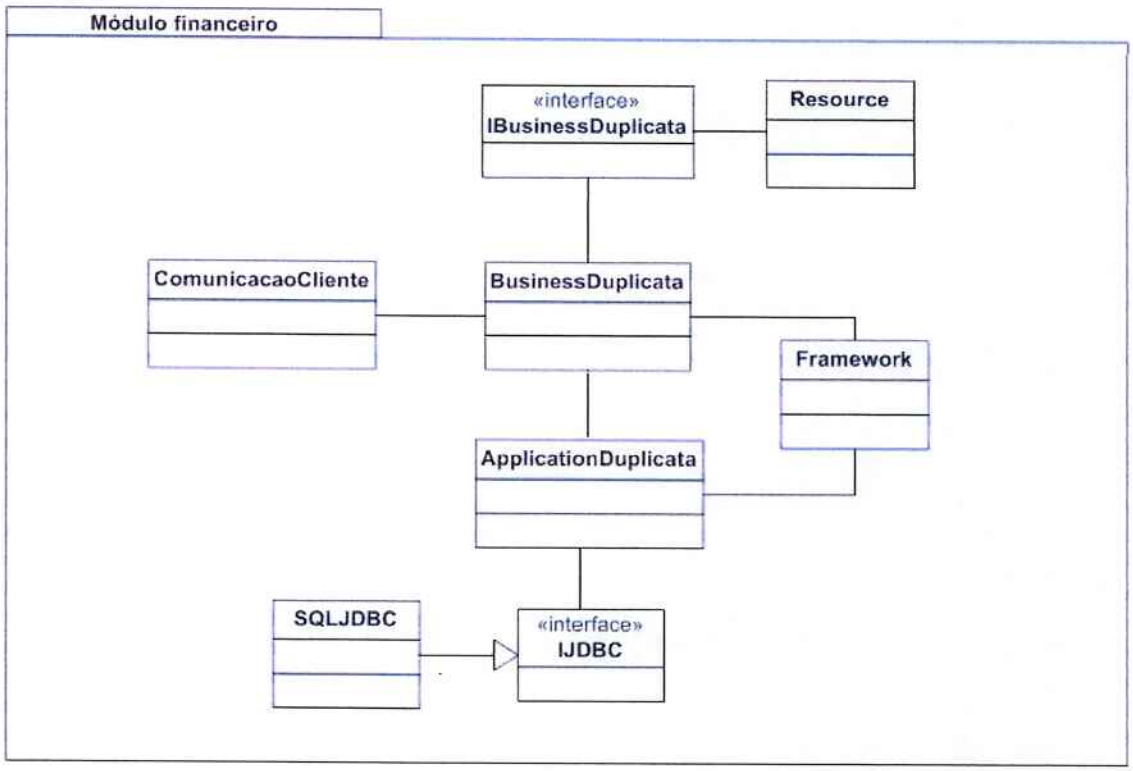

Figura 17: Arquitetura interna dos módulos do projeto

formato XML ou HTML, do servidor dinamicamente. A forma de criação de um servlet é extendendo a classe GenericServlet, que é a classe base de todos os tipos de servlet. No caso específico desse projeto, como utilizamos o protocolo de comunicação HTTP, os servlets são extendidos da classe HTTPServlet.

- **Camada de comunicação com o cliente**: Serviços dessa camada são disparados pelos serviços da camada de negócio para buscarem informações provenientes de outros módulos (por exemplo: um serviço da camada de negócio, que traz informações referentes a uma determinada "conta a pagar", pode solicitar informações sobre a compra que disparou essa conta realizando uma conexão com o módulo de compras, através de uma comunicação cliente para o módulo de compras, utilizando serviços dessa camada).

- **Camada de negócio**: Essa camada disponibiliza seus serviços através de uma interface, ela se caracteriza por possuir serviços de alta granularidade que contém as regras de negócio do sistema.

- **Camada de aplicação**: Serviços de menor granularidade, encarregados de efetuarem buscas ao banco de dados e para fornecer as informações necessárias para a camada de negócio, que trata essas informações de forma a acrescentar as regras de negócio do sistema.
- **Framework**: Possui as funcionalidades e classes básicas utilizadas por todos sistema.

Principais funcionalidades:

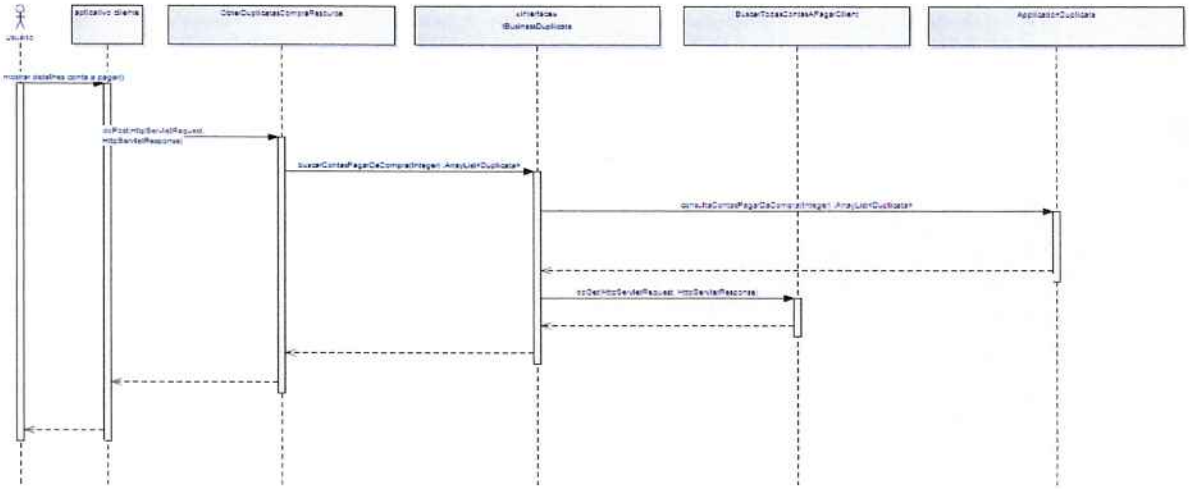

Figura 18: Diagrama de sequência do pagamento de contas

1. Serialização XML através da classe EntityXML.
2. Comunicação cliente para servidores TOMCAT (disponibilizando serviços através do protocolo HTTP com servlets 3.0) através das classes HTTPArguments, RestChannel e ChannelException.
3. Comunicação AJAX crossdomain através das classes CrossDomainFilter e RegisterListener.

O framework também possui um pacote com as entidades do sistema, que são:

1. Cargo
2. CFOP
3. Cliente
4. Compra
5. CompraUnitaria

6. Departamento
7. Duplicata
8. Estoque
9. FormaPagamento
10. Fornecedor
11. Funcionario
12. Grupo
13. GrupoProduto
14. MovimentacaoEstoque
15. Prioridade
16. Produto
17. RequisicaoProduto
18. StatusVenda
19. StatusCompra
20. TipoProduto
21. Transportadora
22. Usuario
23. Venda
24. VendaUnitaria

Observação: *A especificação das classes citadas acima encontra-se em anexo.*

A figura ilustra o diagrama do módulo financeiro, ele foi escolhido por ser um dos módulos mais simples.

Os outros módulos apresentam a mesma arquitetura porem com um aumento do número de serviços e da complexidade da composição dos mesmos.

Os serviços de podem ser mapeados em diagramas de sequência para facilitar seu entendimento e e ter uma idéia mais clara de como será o fluxo de dados.

Todos os outros serviços podem ser representados em um diagrama de sequência semelhante ao exemplo, porém com uma complexidade maior.

4.3.4 Comunicação segura

Em sistemas ERP a segurança é uma parte vital do sistema, pois está se lidando com dados confidenciais das empresas. Por isso, os dados de uma empresa não podem ocupar o mesmo espaço físico do que os dados de outra. Nosso projeto garante essa separação, pois cada empresa usará um sistema com seu próprio banco de dados e módulos na nuvem separado do sistema de uma empresa. Se comportando assim, como se cada empresa tivesse seu sistema instalado em um data center isolado.

Quanto a segurança do tráfego de dados na nuvem e na Internet, primeiramente foram caracterizados dois modos de transferência de dados entre módulos, *a comunicação com o cliente* e a *comunicação lateral*, entre módulos distintos. Como já discutido, o intermédio entre componentes arquiteturais é feito pelo Diretório de Serviços, para garantir baixo acoplamento. Torna-se necessário saber dinamicamente a localização do mesmo, a fim de garantir o funcionamento correto da aplicação e prover a inerente irretratabilidade, confidencialidade e autenticidade nesta comunicação. Há, para tanto, uma página segura por módulo para a definição de endereços do diretório de serviço, e as comunicações entre módulos distintos é realizada por meio de HTTPS, com certificados em ambos os lados, garantindo a segurança da *comunicação lateral*.

Uma forma de fazer a conexão segura entre o cliente e o lado do servidor é utilizando SSL VPN (Secure Socket Layer virtual private network), que utiliza VPN com os web browsers padrões sem necessitar instalar software no computador do cliente, segundo [38] A idéia é utilizar uma conexão VPN com o serviço disponível na nuvem e utilizar um protocolo SSL para manter a conexão segura. O protocolo SSL é utilizado pela maioria dos browsers atuais.

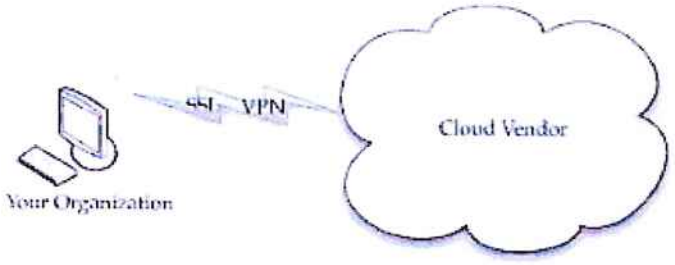

Figura 19: Conexão entre cliente e a nuvem usando VPN e SSL

Claro que para manter a segurança do acesso os usuários devem seguir normas de segurança e os computadores devem ter antivírus e softwares atualizados e seguros.

Tem-se que destacar que a empresa que oferece o serviço de computação em nuvem deve ter uma infra-estrutura segura, onde os dados não fiquem expostos de maneira indevida e que tenha mecanismos de redundância para que não ocorra a perda de dados.

4.3.5 Resultados

Para realizar os testes de desempenho de nossa aplicação foi selecionada a página de lista de formas de pagamento do módulo de compras. Foi criada uma réplica dessa página em JSP para ser possível a comparação entre a utilização de javascript com requisições assíncronas para a montagem da página, que é o caso de nosso projeto, e a página já ser montada do lado do servidor e depois carregada, que é o caso da página JSP.

Testes de carga foram realizados utilizando o site browsermob.com, que apesar de ser um serviço pago, permite realizar alguns testes gratuitos. Ele utiliza a nuvem para simular acessos múltiplos à página a ser testado. Ele também faz todo um levantamento estatístico para demonstrar, ao fim do teste, o desempenho da página. Os testes foram realizados para as duas páginas para se fazer uma análise dos resultados.

Pelos resultados pode-se observar que nas páginas carregadas dinamicamente por javascript o tempo de carregamento é menor em relação à página JSP. Um tempo de resposta rápido acrescenta positivamente a experiência do usuário, pois ele espera o feedback o mais rápido possível de suas ações.

4.4 Considerações finais

Este capítulo prestou-se a apresentar a solução proposta, agregando todos os conceitos dos capítulos anteriores e findando com testes sobre a solução proposta, a fim de verificar sua adequação aos objetivos visados.

5 Conclusão

"Ideas are like rabbits. You get a couple, learn how to handle them, and pretty soon you have a dozen."

John Steinbeck

Como resultado majoritário deste trabalho foi possível averiguar, na prática, o impacto da modelagem dos processos de negócio no mapeamento de serviços final, adequando-se, porém, à arquitetura projetada.

Notou-se, mediante a testes e ao uso do sistema, ganhos em termos de flexibilidade devidos à escolha de padrões web, no sentido que são muitas as ferramentas de auxílio ao desenvolvimento e procura de bugs que apresentam suporte aos serviços e telas projetados, bem como plataformas em nuvem que apresentem suporte sem necessidade de grandes transições tocante a softwares.

Há, por fim, que se destacar que o desenvolvimento de um middeware robusto e flexível figura como uma das etapas mais impactantes no decorrer de um projeto que siga a mesma linha, havendo espaço para o desenvolvimento de uma interface de desenvolvimento padronizada e visual (IDE) como muitas existentes no mercado para a criação de soluções otimizadas e que promovam um entendimento comum às áreas de TI e negócios, na forma de fluxogramas, por exemplo, e que façam uso do mesmo, pois, uma vez adotados um conjunto comum de padrões por parte da organização, é possível padronizar ferramentas de desenvolvimento de software em conformidade com a padronização de processos, promovidas por muitos modelos de maturidade.

5.1 Trabalhos futuros

Muitas foram as idéias que surgiram durante o desenvolvimento deste trabalho. Algumas delas nos levaram a ruas sem saída, ao nível de abandonar soluções completas e iniciar o trabalho do zero (mais de uma vez :), enquanto outras abriram portas para lugares inexplorados que mudaram nosso modo de pensar e que, por sua vez, nos levaram a mais idéias - de modo cíclico. Obviamente não cremos que um trabalho, em grande parte, criativo, chegue ao fim - motivo pelo qual não fomos capazes de dar cabo a todas as melhorias que desejávamos. Deixamos, entretanto, um subconjunto destas idéias para possíveis trabalhos futuros:

- **Serialização binária e otimizada**: A comunicação cliente-servidor ocorre por meio de protocolos não otimizados (XML ou JSON). Tais protocolos não são adequados para as aplicações corporativas por apresentarem alto consumo de banda na transferência de dados entre cliente e servidor, gerando custos de infraestrutura. Possíveis exemplos de protocolos mais adequados seriam o *MessagePack*, *Hessian* ou o *Protocol Buffers*.

- **Fluxo de mídia P2P por meio de padrões WEB**: A maioria das organizações demandam sistemas que permitam chat entre usuários, muitas vezes integrando som e vídeo. No caso de um chat simples poderíamos fazer uso de padronizações recentes, tais como o Web Socket [7]. Já quanto a vídeo e audio seria interessante ver uma implementação da tag VIDEO do HTML5 (quem sabe usando o recente formato de vídeo aberto: WebM [6]?) integrada em um sistema ERP. Um último elemento interessante seria a interação entre funcionários distantes geograficamente por meio de um fluxo de mídia P2P (futuro HTML Device [1]) utilizando algum mecanismo que permita a integração de dispositivos de captura de mídias (tal como a HTML Media Capture [2]). Seria possível?

- **Internet of Services e composição de serviços**: Seria interessante realizar uma composição dos serviços de modo gráfico no sentido de se aproximar do cenário de Internet of Services e permitindo que os desenvolvedores documentem uma aplicação de modo facilitado, bem como a desenvolvam em conjunto com os especialistas no negócio.

- **Web of Things e ubiquidade**: Fazer mais testes tocante a ubiquidade de dispositivos e permitir que módulos de um sistema ERP residam nos mesmos seria interessante em muitos contextos. Medição de dados de uma rede de etiquetas RFID contidas em um estoque, por exemplo.

5.2 Contribuições de trabalho

De modo a identificar os autores dos diversos produtos de trabalho que culminaram no projeto em questão, apresenta-se, a seguir, as contribuições de trabalho dos membros da equipe:

- Professor Dr. Jorge Risco Becerra
  - Gerenciamento do projeto
  - Estabelecimento de metas para a equipe
  - Validação e correção da monografia
  - Correção e validação da apresentação
  - Orientação, devido a um amplo conhecimento, nas áreas de
    * Engenharia de software
    * Arquitetura de sistemas
    * Sistemas distribuídos
    * Cloud computing

    * SOA

- Ricardo Moreira Muniz
  - Definição do modelo de arquitetura distribuída
  - Pesquisa sobre as tecnologias que se adequariam ao projeto
  - Projeto do middeware da comunicação lateral e vertical
  - Escolha da nuvem adequada para o sistema
  - Desenvolvimento do protótipo
  - Escrita da monografia
- Alexandre Del Nero
  - Análise dos processos de negócio
  - Definição dos serviços SOA
  - Desenvolvimento da arquitetura interna dos módulos do ERP
  - Desenvolvimento do protótipo
  - Escrita da monografia
- Danilo Elias
  - Estudo e análise conceitual sobre cloud computing
  - Projeto do banco de dados
  - Definição da nuvem adequada para o sistema
  - Desenvolvimento do protótipo
  - Escrita da monografia

5.3 Considerações finais

O que o futuro reserva para os Sistemas de Gestão Empresarial? Como de praxe, não há uma resposta simples, pois o que pode ser bom para uma empresa pode não ser bom para outra.

É interessante olhar como as tecnologias desses sistemas evoluíram no decorrer do tempo e se utilizar desta evolução como ponto de partida para ter idéia de como este mercado estará posicionado nos próximos anos. Uma coisa é certa: empresas dependentes de grandes pacotes de gestão empresarial caminham lentamente atreladas as suas dependências, sendo difícil e oneroso

se afastar deste modelo de negócios - que possui, dentre outros possíveis exemplos, atualizações obrigatórias e contratos de manutenção custosos.

Há, por outro lado, soluções mais flexíveis para as empresas, tais como soluções que se baseiam em SaaS (cloud computing) e que oferecem maior liberdade e flexibilidade para uma gama cada vez maior de empresas a custos acessíveis (tenha em vista o custo dos sistemas de gestão tradicionais), criando um modelo particularmente atraente, agora, para empresas mais ágeis.

As pequenas e médias empresas, por não estarem atreladas a contratos e tecnologias milionárias, possuem mentes abertas para implantarem soluções flexíveis, e que, hoje, possuem custos progressivos e integração facilitada, dado que se embasam em padrões, permitindo ubiquidade entre dispositivos - outrora impossível ou custoso com os softwares tradicionais.

De qualquer modo, a evolução nos sistemas de gestão rumos a inovação e flexibilidade são certos, frente a crescimentos esperados de cerca de 650% na quantidade de dados empresariais que serão gerados nos próximos anos, segundo pesquisas recentes do Garter Group, ou mesmo a crescente gama de dispositivos que fazem o intermédio entre o computador e o celular, e que clamam seu espaço na empresa competitiva de hoje. [32]

6 Anexos

6.1 Processo de negócio principal

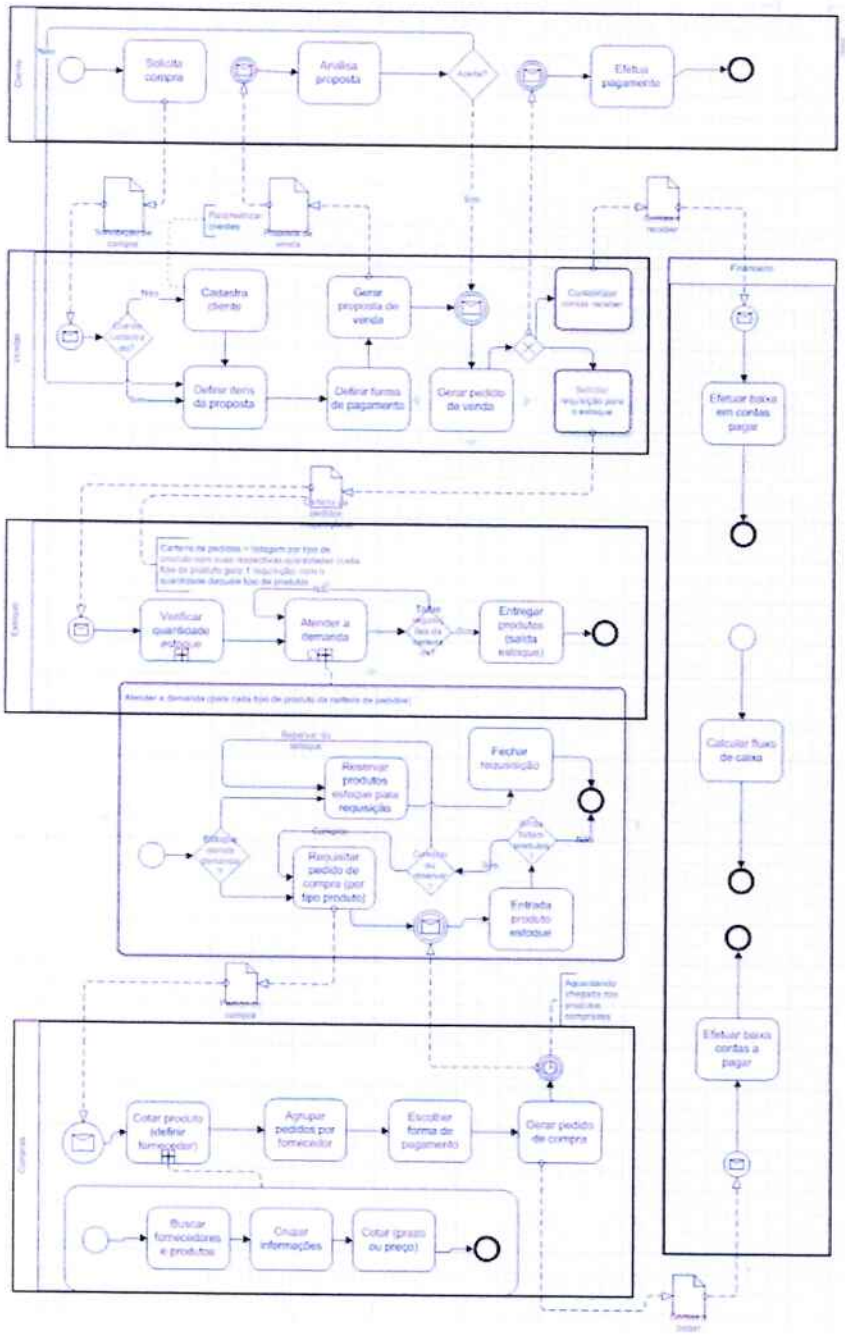

Figura 20: Modelagem do processo de negócio principal do sistema

7 Bibliografia

[1] Html device. http://dev.w3.org/html5/html-device/. 41

[2] Html media capture. http://www.w3.org/TR/capture-api/. 41

[3] Open bravo erp. http://www.openbravo.com/. 5, 25

[4] Serviço brasileiro de apoio às micro e pequenas empresas (sebrae). http://www.sebrae.com.br/. 2, 4

[5] Weberp. http://www.weberp.org. 5, 23, 24

[6] The webm project. http://www.webmproject.org/. 41

[7] Websocket api. http://dev.w3.org/html5/websockets/. 41

[8] Apache. Ofbiz: The apache open for bussiness project. http://ofbizguru.com/. 5, 23

[9] Nicholas G. Carr. It doesn't matter. *Harvard Business Review*, 2003. 11

[10] CloudTweaks. Cloud computing – demystifying saas, paas and iaas, Acessado em 2 de Dezembro de 2010. http://www.cloudtweaks.com/2010/05/cloud-computing-demystifying-saas-paas-and-iaas/. 6

[11] Schulz Consulting. What is saas, cloud computing, paas and iaas?, Acessado em 2 de Dezembro de 2010. http://www.s-consult.com/2009/08/04/what-is-saas-cloud-computing-paas-and-iaas/. 6

[12] Juliano Correa. Adoção, seleção e implementação de um erp livre, 2008. 21

[13] Instituto Brasileiro da Qualidade e Produtividade. Empreendedorismo no brasil 2009, 2009. 4

[14] Felipe Martins da Silva, Fernando Raffani, and João Paulo Seabra Santos. *Desenvolvimento de Software ERP Usando SOA e BPM aplicado ao Programa de Educação Continuada da Escola Politécnica*. Projeto de formatura, Escola Politécnica da Universidade de São Paulo, 2006.

[15] Shahin Dezdar and Ainin Sulaiman. Successful enterprise resource planning implementation: taxonomy of critical factors. *Industrial Management & Data Systems*, 109(8):1037–1052, 2009. 7

[16] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology and Design*. Prentice Hall, 2005.

[17] Thomas Erl. *SOA Design Patterns*. Prentice Hall, 2009. 14, 17

[18] Muscatello et al. Implementing enterprise resource planning (erp) systems in small and mid-size manufacturing firms. *International Journal of Operations & Production Management*, 233(8):850–871, 2003. 2, 7

[19] Vinícius Evandro. Erps open source. http://www.vivaolinux.com.br/artigo/ERPs-Open-Source-cipais-solucoes?pagina=3. 5, 22

[20] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, 2000. 13

[21] Sara Grant. Confronting the reality of web services: Q&a with andrew p. mcafee. *Harvard Business School Working Knowledge*, 2005. http://hbswk.hbs.edu/item/4800.html. 19, 21

[22] Mark D. Hansen. *SOA using Java Web Services*. Prentice Hall, 2007. 16

[23] Cloud Computing Info. Defining cloud computing, Acessado em 2 de Dezembro de 2010. http://clouddb.info/2009/02/23/defining-cloud-computing-part-6-iaas/. 6

[24] IDC Manufacturing Insights. Beating complexity, achieving operational excellence, 2010. 5, 9, 10, 11

[25] Till Janner and Christoph Schroth. Web 2.0 and soa: Converging concepts enabling the internet of services, 2007. http://www.computer.org/portal/web/buildyourcareer/fa008. 5, 18, 19, 20

[26] Marcos Antonio Koteski. As micro e pequenas empresas no contexto econômico brasileiro, 2004. 2, 4

[27] Joe McKendrick. Managing all the data in the world, 2005. http://www.zdnet.com/blog/service-oriented/managing-all-the-data-in-the-world/499. 18

[28] Joe McKendrick. Soa cuts it budget by half, 2005. http://www.zdnet.com/blog/service-oriented/soa-cuts-it-budget-by-half/259. 18

[29] Joe McKendrick. Soa's strong medicine, 2005. http://www.zdnet.com/blog/service-oriented/soas-strong-medicine/254. 18

[30] Lei Complementar nº 123. Estatuto nacional da microempresa e empresa de pequeno porte, 2006. http://www.planalto.gov.br/CCIVIL/LEIS/LCP/Lcp123.htm. 4

[31] OASIS. Oasis soa reference model tc. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm. 17

[32] Julie Pitta. Divining the future of erp software, 2010. http://content.dell.com/us/en/corp/d/large-business/future-of-erp-software.aspx. 43

[33] Robin Poston and Severin Grabski. Financial impacts of enterprise resource planning implementations. *International Journal of Accounting Information Systems*, pages 271–294, 2001. 4, 7, 8

[34] SaaSBlogs. Demystifying the cloud: Where do saas, paas and other acronyms fit in?, Acessado em 2 de Dezembro de 2010. http://www.saasblogs.com/2008/12/01/demystifying-the-cloud-where-do-saas-paas-and-other-acronyms-fit-in/. 5, 6

[35] Michael Stonebraker. The case for shared nothing. *Database Engineering*, 9(1), 1986. 29

[36] William E. Sullivan and Kweku-Muata Bryson. Designing effective incentive-oriented contracts for application service provider hosting of erp systems. *Business Process Management Journal*, 9(6):705–721, 2003. 5

[37] Nelma Terezinha Zubek Valente. Implementação de erp em pequenas e médias empresas: Estudo de caso em empresa do setor da construção civil, 2004. 8, 9

[38] Toby Velte, Anthony Velte, Toby J. Velte, and Robert C. Elsenpeter. *Cloud Computing: A Practical Approach*. Mc-Graw Hill, 2009. 38

[39] W3C. http://www.w3.org/. 16

[40] W3C. Web services glossary, 2004. http://www.w3.org/TR/ws-gloss/. 13

[41] Fabio Eiji Yamasaki, Marcelo Massayoshi Takizawa, and Marcelo Toshio Uenoyama. *Soluções em Business Intelligence baseadas em Software Livre*. Projeto de formatura, Escola Politécnica da Universidade de São Paulo, 2005.