SÉRGIO MICELLI FILHO

ARQUITETURA DE UMA PLATAFORMA DE MICROPAGAMENTOS BASEADA EM SOA E CONCEITOS DE AUTONOMIC COMPUTING

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de MBA em Engenharia de Software

São Paulo
2007

SÉRGIO MICELLI FILHO

ARQUITETURA DE UMA PLATAFORMA DE MICROPAGAMENTOS BASEADA EM SOA E CONCEITOS DE AUTONOMIC COMPUTING

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de MBA em Engenharia de Software

Área de Concentração:
Engenharia de Software

Orientador:
Prof. Dr. Jorge Risco

São Paulo
2007

FICHA CATALOGRÁFICA

Micelli Filho, Sérgio
Arquitetura de uma plataforma de micropagamentos baseada em SOA e conceitos de *autonomic computing* / S. Micelli Filho. -- São Paulo, 2007.
86 p.

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

1.Engenharia de software 2.Arquitetura de software 3.Arquitetura orientada a serviços I.Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II.t.

DEDICATÓRIA

Aos meus pais, minha esposa e meu querido filho Arthur.

ok

AGRADECIMENTOS

Ao professor Jorge Risco, pela orientação e pelo constante estímulo transmitido durante o trabalho.

A minha esposa e meu filho Arthur, pela paciência.

RESUMO

O trabalho pretendeu investigar de que forma o modelo de referência SOA (Service-Oriented Architecture) e conceitos de Autonomic Computing (AC) podem influenciar e contribuir para arquitetura de uma plataforma de micropagamentos. Ele analisou as características deste tipo de sistema/negócio, fundamentos, vantagens e desvantagens, métodos e processos de implementação de uma solução SOA e de Autonomic Computing, e como SOA e AC interagem e se relacionam arquiteturalmente. São apresentadas conclusões e sugestões sobre a aplicação de SOA e AC em um sistema de micropagamentos bem como sua extensão para outras categorias de aplicação.

Palavras-chave: Engenharia de software, Arquitetura de software, Arquitetura orientada a serviços (SOA), Autonomic Computing e Micropagamentos.

ABSTRACT

This monograph intends to investigate how SOA Reference Model and Autonomic Computing concepts can influence and contribute to a micropayments platform architecture. It analyses the business and the foundations, advantages and disadvantages, methods and implementation process of a SOA and Autonomic Computing solution, and how SOA and AC can work together in architecture. Conclusions and suggestions are presented about SOA and AC appliance in a micropayments system and extension to other applications.

Keywords: Software Engineering, Software Architecture, Service Oriented Architecture (SOA), Autonomic Computing e Micropayments.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O crescimento da oferta de serviços de baixo montante e seu consumo por diferentes canais e dispositivos tecnológicos têm aumentado a demanda por sistemas de micropagamentos, que em contrapartida, têm se mostrado difíceis de viabilizar, face aos altos custos operacionais versus as baixas margens impostas aos negócios deste segmento.

Por outro lado, dois novos conceitos e tecnologias recentes e complementares têm surgido para tentar otimizar a utilização de recursos e as respostas ao dinâmico ambiente de negócios atual: a *SOA* (*software-oriented architecture* ou arquitetura orientada a serviços) e a *AC* (*autonomic computing* ou computação autônoma).

Esses dois conceitos são complementares e ao mesmo tempo têm pontos em comum que farão parte da investigação deste trabalho.

1.1 Motivação

O presente trabalho foi concebido como uma oportunidade de aplicação dessas duas tecnologias/conceitos em um *case* real de negócio: um sistema de micropagamentos.

Tais sistemas apresentam um excelente cenário para aplicação dessas tecnologias, seja pelos aspectos financeiros relacionados ao aumento da demanda ou pela necessidade de melhorias técnicas para viabilização comercial dos mesmos, visando:

- aumento do ROI (*return on investment* ou retorno do investimento)
- redução do TCO (*total cost of ownership* ou custo total de propriedade)
- aumento do QoS (qualidade do serviço)

Aplicar a arquitetura *SOA* e conceitos de *Autonomic Computing* em conjunto, visando a geração de uma plataforma de micropagamentos flexível, autogerenciada, escalável e de baixo custo, que possa ser implementada e viabilizada comercialmente, é a grande motivação deste trabalho.

1.2 Objetivos

O objetivo deste trabalho é apresentar uma contribuição para a arquitetura lógica de uma plataforma de micropagamentos baseada na estratégia e estilo arquitetural *SOA* (arquitetura orientada a serviços) e conceitos de *Autonomic Computing*, utilizando um conjunto de processos-chave de negócio (cenário proposto).

A ênfase na arquitetura deve-se principalmente por sua associação com muitos riscos de projeto, principalmente no desenvolvimento da primeira geração de uma aplicação [Kroll; Kruchten, 2003, p. 38].

O modelo arquitetural será construído e documentado utilizando estilo arquitetural SOA, em abordagem top-down, com foco no modelo de negócio, pois inexistem ativos de software.

1.3 Abrangência

A intenção deste trabalho não é discutir conceitos, ideologias ou a aplicabilidade de sistemas de micropagamentos em soluções de comércio eletrônico, mas sim, analisar as contribuições da arquitetura *SOA* e da *Autonomic Computing* nesses tipos de sistema para viabilizá-los segundo alguns critérios.

O desafio deste trabalho é consolidá-los em uma única visão de aplicação da tecnologia AC para uma plataforma de micropagamentos baseada em SOA.

Para isso, serão destacados até três processos de negócio que se enquadrem nos objetivos do projeto – potencializando características da arquitetura SOA e AC.

1.4 Metodologia de trabalho

A metodologia utilizada baseou-se na pesquisa bibliográfica e documental dos temas abordados, principalmente em artigos, revistas técnicas e *sites* específicos, sendo:

- levantamento bibliográfico em bases de dados nacionais e internacionais a partir de palavras-chave em português e inglês;
- busca pelos textos completos dos trabalhos selecionados no levantamento;
- resumo analítico dos trabalhos relevantes;
- cruzamento de informações e citações;
- redação da revisão e citações, conforme assunto e tópicos abordados.

Para geração do modelo arquitetural, objeto deste trabalho, foram criadas três frentes de pesquisa:

- conceitos, sistemas e plataformas de micropagamentos
- conceitos e arquiteturas SOA e AC, isoladamente
- arquiteturas SOA e AC, conjuntamente

Para embasamento teórico relacionado a conceitos, sistemas e plataformas de micropagamentos foi realizado levantamento destacando-se artigos de diversos autores (Ex.: Rivest, Szabo, Odlyzko etc) e estudos de caso de empresas (Ex.: Bitpass, Payloadz, Firstgate, Paystone, Peppercoin, PaymentOne, Cybercash, e-Cash).

Os tópicos relacionados a micropagamentos abordados para elaboração e desenvolvimento do trabalho foram:

- definição e conceitos de micropagamentos
- micropagamentos e o comércio eletrônico
- conteúdo de valor agregado e comoditizado
- demanda por sistemas de micropagamentos
- modelos de receita para micropagamentos
- sistemas de micropagamentos no contexto de sistemas de pagamentos
- modelos de pagamento
- modelos de sistemas de micropagamentos
- desafios tecnológicos dos sistemas de micropagamentos
- casos de sistemas de micropagamentos
- macro-desafios de sistemas de micropagamentos
- produtos e serviços candidatos a micropagamentos

Os conceitos e arquiteturas *SOA* e *Autonomic Computing* foram estudados isoladamente e conjuntamente (dos conceitos básicos para arquitetura), principalmente através de artigos disponibilizados pela IBM, dos quais destacam-se:

SOA:

- *SOA Foundation - An Architectural Introduction and Overview*
- *SOA Foundation - Business Process Management Scenario*
- *SOA Foundation - Service Creation Scenario*
- *Modeling Service-Oriented Solutions*
- *Service-oriented modeling and architecture (SOMA)*

Autonomic Computing:

- *The Vision of Autonomic Computing (IEEE)*
- *Autonomic Computing: IBM's perspective on the state of IT*
- *An architectural blueprint for Autonomic Computing*

Os tópicos abordados no tema Autonomic Computing foram:

- evolução tecnológica e complexidade
- aumento da demanda por serviços de TI e restrições de oferta
- novas abordagens para solução do problema da complexidade
- exemplos históricos relacionados a automação
- autonomic computing como solução para o problema da complexidade
- campos de estudo adjacentes e complementares a AC
- elementos-chave de sistema computacionais autônomos
- características self-CHOP
- potencialidades da AC
- níveis de maturidade de ambientes AC
- desenvolvimento de sistemas computacionais autônomos
- arquitetura de sistemas computacionais autônomos
- componentes de sistemas computacionais autônomos
- ciclo de vida de sistemas computacionais autônomos
- exemplos de aplicação da AC

O objeto deste trabalho, que consolida a arquitetura SOA com a abordagem arquitetural da *Autonomic Computing*, teve como principais fontes:

- *white-paper* IBM "*Autonomic Computing: Enabling Self-Managing Solutions*"

- artigo IBM "*Combine autonomic computing and SOA to improve IT management*", de Christine Draper, IBM Senior Technical Staff Member, 2006

A implementação AC visou o nível 5 de maturidade ("*Dynamic business-policy-based management*").

1.5 Organização

O capítulo 1, "Introdução", apresenta a motivação, objetivos, abrangência e metodologia de trabalho utilizadas.

O capítulo 2, "Revisão da literatura", apresenta a teoria básica sobre micropagamentos, *autonomic computing* e arquitetura orientada a serviços (SOA), que dará a sustentação necessária para apreciação dos capítulos seguintes.

O capítulo 3, "Solução proposta", apresenta o roteiro que será seguido para implementar a arquitetura.

O capítulo 4, "Desenvolvimento da arquitetura", apresenta as etapas de construção da arquitetura SOA / AC.

O capítulo 5, "Resultados e discussão", apresenta a consolidação do capítulo 4 em uma visão resumida da arquitetura.

O capítulo 6 apresenta as conclusões do trabalho.

2 REVISÃO DA LITERATURA

Como base fundamental para o desenvolvimento da solução proposta, foram pesquisados, estudados e analisados três assuntos: sistemas de micropagamentos, *autonomic computing* e *SOA*.

Ao final do capítulo os três assuntos são agrupados em uma única visão -"SOA e AC como solução para micropagamentos" – fundamental para o desenvolvimento do corpo do trabalho.

2.1 Micropagamentos

Micropagamento é uma transação de comércio eletrônico de baixo valor. Pode referir-se ao débito de alguns centavos ou uma fração de centavo por uma transação. Pode também se referir a consolidação de diversas aquisições de pequeno valor e débito em cartão de crédito ao final do dia ou período pelo montante total[http://computing-dictionary.thefreedictionary.com/Micropayments].

Uma outra definição de micropagamento refere-se ao método de pagamento utilizado por comerciantes que oferecem produtos, serviços ou informação por alguns centavos ou alguns reais - processar esses pequenos pagamentos como em processos tradicionais reduziria significativamente o montante de lucro ganho com a venda [http://www.merchantseek.com/glossary.htm].

Pode-se também entender micropagamento como uma forma de transferir dinheiro em situações onde receber dinheiro através de sistemas de pagamento tradicionais é impraticável ou muito caro pelo montante sendo recebido – muitas vezes sistemas de micropagamentos acumulam muitos micropagamentos e recebem o montante acumulado como um único pagamento, antes ou depois das transações.

Em resumo, sistemas de micropagamentos têm o objetivo de proporcionar a venda de conteúdo digital de baixo valor unitário de forma fluída, rápida e segura para o comprador e rentável para o provedor/vendedor - contudo, a menos que se tenha um acúmulo mínimo de itens vendidos, o custo de processamento de cada transação tende a ser muito maior do que a receita obtida.

Apesar do grande sucesso mundial do comércio eletrônico para itens de consumo tradicionais como CDs, DVDs, livros, passagens aéreas e outros produtos e serviços, há uma grande demanda ainda não atendida para comercialização de produtos e serviços de baixo montante, mas alto valor agregado através de micropagamentos: músicas, vídeos ppv (*pay-per-view*), artigos em revistas e jornais, quadrinhos, acessos a bancos de dados etc. Essa demanda reflete muitas vezes a sobreposição do conteúdo de valor agregado percebido sobre o conteúdo comoditizado - uma antiga discussão relacionada à validade dos micropagamentos - se pessoas pagariam pequenos montantes por conteúdo online.

Os críticos dos micropagamentos tipicamente apontam o fracasso das iniciativas no final da década de 90 - os primeiros provedores de micropagamentos como *Beenz, Digicash e Flooz* - quando a internet ainda estava decolando, o conteúdo era fortemente subsidiado e as pessoas tinham restrições a gastar dinheiro online – micropagamentos ou qualquer outra forma de pagamento. Em resumo, a maioria dos comerciantes online não estavam interessados em vender itens de baixo montante e os usuários esperavam que o conteúdo online fosse grátis.

Desde que o volume de pagamentos online cresceu dramaticamente, muito do conteúdo digital é pago atualmente via assinaturas ou propaganda, e os internautas já não ficam mais surpresos de serem cobrados.

Alguns modelos atuais de receita são:

- conteúdo grátis (não é modelo de receita em si)
- propaganda
- assinaturas
- carrinho de compras
- micropagamentos

Para ilustrar a demanda atual por micropagamentos: a *Apple* vendeu mais de 100 milhões de músicas a US$ 0,99 nos primeiros anos de operação do serviço *iTunes*.

A habilidade em transformar a internet em uma fonte de lucratividade não depende apenas da boa vontade de seus usuários para pagar por serviços especiais, mas também da habilidade para fazê-lo facilmente e rapidamente quando solicitados - muitos produtos online poderão ser vendidos se os internautas dispuserem de um mecanismo simples e automático para pequenos montantes.

Os sistemas de pagamento online existentes, dominados por operadoras de cartão de crédito, falham nesse sentido por uma série de razões: cartões de crédito são inúteis para transações individuais de muito baixo valor, indo de alguns centavos a alguns reais, pois seus custos "engolem" os lucros dos vendedores. Além disso, consumidores e comerciantes pagam altas taxas de juros e transacionais para os emissores de cartão de crédito. A dominância de cartões de crédito na internet deve-se a ausência de melhores alternativas. Uma melhor alternativa para pagamentos online é urgente.

O processo de pagamento – identificação, avaliação de crédito, processamento da transação – também é algo impraticável para micropagamentos.

O sistema de pagamento escolhido para *websites* depende do tamanho e abrangência da audiência e o valor do conteúdo para audiência. Quanto maior a audiência e mais valioso o conteúdo, maior receita pode ser esperada e por extensão, tempo e dinheiro investidos em uma infra-estrutura de pagamentos.

Existem diversos modelos de pagamento a escolher. Cada modelo de pagamento tem seus próprios prós e contras, e nenhum modelo necessariamente é o melhor para todas as situações.

Os modelos de pagamento praticados atualmente são:

- comprador → serviço de micropagamento → comerciante
- comprador → comerciante

Tipicamente, compradores adquirem conteúdo digital de baixo montante de duas formas: realizando um depósito pré-pago para um único provedor para cobrir diversas aquisições ou recebendo a conta do provedor após uma série de pequenas aquisições. Esses modelos prendem o usuário a determinado provedor e necessitam que o mesmo mantenha um processo de rastreamento de todas as aquisições individuais.

Provedores de micropagamentos geram receitas através da cobrança de taxas dos provedores de conteúdo. Eles operam com empresas de hospedagem para proporcionar confiabilidade e estabilidade para suas plataformas: fontes de energia contínuas, geradores de energia backup, fontes redundantes de acesso a internet e outros tipos de redundância.

Os desafios chave para a tecnologia de micropagamentos são segurança, facilidade de uso, habilidade de manipular altos volumes transacionais e armazenar registros relativos a regulamentações bancárias.

Vendedores usam senhas e encriptação para gerar segurança para os micropagamentos; aplicações de micropagamentos normalmente não requerem que usuários instalem *plug-ins* ou outros softwares, que costumam ser fontes de brechas de segurança.

Sistemas de micropagamentos também trabalham com tecnologia de checagem de erros para reduzir as chances de erros na manipulação de transações – muitos provedores oferecem facilidade de uso através da utilização de interfaces parecidas com as de cartões de crédito.

As tecnologias comerciais de micropagamentos são parecidas em alguns aspectos e diferentes em outros.

Algumas empresas com soluções para micropagamentos são: *Bitpass, Firstgate, Payloadz, Paystone* e *Peppercoin*.

A *Bitpass* deduz o custo de pequenas aquisições da conta do comprador anteriormente pré-carregada (paga) através de cartão de crédito ou *paypal*. Os cerca de 1000 provedores que trabalham com o *Bitpass* registram seus conteúdos e serviços com a empresa e instalam *gateways* em seus servidores. Quando usuários clicam no conteúdo no site do provedor de conteúdo, o sistema pede uma senha em uma nova janela. Uma vez confirmado saldo suficiente, o usuário pode acessar o conteúdo. A *Bitpass* paga os provedores via *paypal* ou transferência eletrônica de fundos.

A *Firstgate* opera com 3000 provedores de conteúdo. Tem 2,5 milhões de consumidores, a maioria na Europa. Para segurança, usam senhas, protocolo/encriptação SSL e gravam o IP dos compradores. Cobram taxas de *setup*, integração e consultoria dos provedores. Os compradores são cobrados via cartão de crédito ou débito, ou conta telefônica, após acumularem alguns dólares em cobrança. Clientes nos EUA podem ser debitados direto de suas

contas bancárias. A *Firstgate* agrega micro-aquisições de consumidores pelos provedores de conteúdo, possibilitando um único processamento de múltiplas transações, reduzindo custos de processamento, taxas de cartão de crédito por compra e custos de administração. A *Firstgate* envia pagamentos para os provedores via cheque ou transferência bancária.

O *Payloadz* opera com 1000 comerciantes e gera 3 milhões em receitas por ano. Usuários pagam os provedores de conteúdo diretamente via *paypal*. Os comerciantes pagam ao *Payloadz* uma taxa mensal baseada nos níveis de venda, ao invés de um percentual por transação. Os comerciantes integram-se ao *paypal*. Usuários finais nunca vêem uma interface *Payloadz*. Utiliza SSL com encriptação RC4 128-bit.

No *Paystone* usuários na América do Norte, Austrália, Nova Zelândia e partes da Europa podem acessar conteúdo *web* após conFigurar uma conta pré-paga através da transferência de fundos para a empresa via boleto ou depósito em dinheiro no *Bank of America*. Comerciantes podem vender conteúdo através da criação de *links* para o sistema de pagamentos – usuários seguem os links, entram com sua senha e são redirecionados para o conteúdo. Segurança via encriptação SSL 128-bit.

No *Peppercoin* compradores dos *sites* participantes vêem a interface *Peppercoin*, similar a uma interface de cartão de crédito. A aplicação não requer pré-cadastro ou pré-depósito de fundos. Compradores entram com a informação do cartão de crédito ou débito e o *Peppercoin* processa a transação. Para segurança é usado o software RSA BSAFE para encriptação e assinatura digital.

Os "macro-desafios" para micropagamentos são:

- Fatores econômicos: micropagamentos estão atrelados a transações de baixo montante e baixa rentabilidade. Pode provar-se não ser economicamente viável para algumas empresas depender de altos volumes transacionais que podem não se materializar.

- Demanda insuficiente: o grande desafio dos provedores de sistemas de micropagamentos é tornarem-se atrativos para os comerciantes e mostrarem-se "pagáveis". Muitos comerciantes querem que um ou dois produtos mostrem-se atrativos e confiáveis para os compradores. Adicionalmente, o conteúdo digital está espalhado pela internet. **Existem poucos sites onde compradores potenciais podem ver grandes quantidades de conteúdo digital agrupados juntos – isso torna menos conveniente comprar material, dessa forma reduzindo o número de micro-aquisições online e a demanda por software/sistemas de micropagamentos**.

O uso de micropagamentos está começando a crescer no Japão e editoras e jornais europeus. Se a tecnologia tornar-se largamente usada, os provedores de soluções deverão defrontar-se com pontos críticos como **reembolso** e **suporte ao cliente**. No futuro deverá haver progresso na interface/usabilidade, e na incorporação de mp em PDAs e celulares – contudo, pesquisadores precisam atentar para como rodar aplicações de micropagamentos em *handhelds*, com limitações de performance, memória e tempo de bateria. Exemplo: a criptografia exigida pelos sistemas de micropagamentos exige considerável poder de processamento, que podem sobrecarregar as baterias de dispositivos móveis. MP devem ter baixa evolução nos próximos anos devido a baixa demanda. Essa é uma tecnologia com grandes ambições e limitada perspectiva. Por outro lado, outros observadores alertam que a tecnologia MP tornar-se-á mais difundida quando mais consumidores pagarem por conteúdo a serviços personalizados.

Existe uma defasagem na cobertura dos sistemas de pagamento para itens de baixo custo que podem endereçar comerciantes com pequenas e grandes audiências. Essa defasagem pode ser coberta por sistemas de micropagamentos. Pontos a destacar:

- Conteúdo grátis e conteúdo coberto por micropagamentos podem resultar em confusão pelo conflito de custo versus valor percebido

- Micropagamentos podem se adequar a pequenas e grandes audiências, independente do tamanho do provedor de conteúdo

- Carrinhos de compra e assinaturas se sobrepõem um pouco na faixa de alto custo de conteúdo, sugerindo similaridades na entrega de conteúdo e métodos de acesso

- Carrinhos de compra e assinaturas podem co-existir com sistemas de micropagamentos e podem possivelmente se beneficiar do compartilhamento ou integração com o modelo de micropagamentos

Criadores de conteúdo tem grande interesse em micropagamentos para auxiliar na comercialização de suas criações – provedores de conteúdo de baixo valor visualizam a tecnologia de micropagamentos como uma forma de gerar receita. Também existe demanda para compra de *ringtones*, ícones e *games* para celulares.

A possibilidade de adquirir produtos e serviços de baixo valor deve eliminar a necessidade de assinaturas de serviços com conteúdos relevantes e irrelevantes, e reduzir a pirataria.

Um dos grandes desafios tem sido convencer usuários e comerciantes a confiarem e adotarem as soluções de micropagamentos.

Enquanto micropagamentos não irão certamente ser utilizados para todas as transações digitais, eles possibilitam:

- Para sites com assinaturas, desenvolver uma nova fonte de receita de usuários que ocasionalmente querem conteúdo ou querem adquirir apenas um ou dois itens;

- Para sites atualmente suportados por propaganda evasiva, oferecer uma versão "não propagandista" do site para aqueles dispostos a pagar uma pequena quantia;

- Para sites que apresentaram conteúdo gratuito no passado, cobrar baixos valores e ainda ganhar dinheiro, pela fuga dos altos custos de processamento do cartão de crédito

- Para sites em geral, cobrar por buscas e outros recursos

- Para qualquer site com conteúdo digital, pagar menos pelo processamento de transações do que pagariam para cartões de crédito

- Para pequenos sites sem orçamento para operação ou suporte programático, cobrar facilmente por seus produtos

- Para sites que querem permanecer gratuitos, adicionar funcionalidade de doação

A opção de micropagamentos pode ser atrativa para provedores com:

- Conteúdo digital (arquivos, musica, jogos, arte, filmes, notícias, software)
- Serviços digitais como buscas especiais ou acesso a um banco de dados
- Assinaturas de conteúdo online como *newsletters*, quadrinhos, musica ou exposições/análises
- Tempo ou recursos insuficientes para programar e manter um sistema de carrinho de compras, sistema de assinatura ou sistema de gerenciamento de conteúdo
- Receita insuficiente para suportar o investimento em um certificado SSL, uma conta comercial ou um sistema de controle de acesso para gerenciar a aquisição e entrega de conteúdo
- Baixo interesse ou mão-de-obra insuficiente para gerenciar os requisitos de processamento de pagamentos de compradores de toda a internet

2.2 Autonomic Computing

O próximo grande desafio da TI é a complexidade.

A indústria de alta tecnologia gastou décadas criando sistemas computacionais para solucionar uma variedade de problemas de negócio.

As mudanças e o crescimento das redes, periféricos e sistemas geraram falhas, incompatibilidades, problemas de hardware e software e erros humanos, que requerem intervenção adicional para reparos, ajustes ou mesmo melhoria de performance.

Atualmente, acima de 40% dos investimentos em TI são utilizados para fazer tecnologias "conversarem", sendo 50% para serviços e produtos que não agregam valor ao negócio. Ao mesmo tempo, busca-se mais do que nunca o aumento do ROI (*return on investment*), a redução do TCO (*total cost of ownership*) e melhoria no QoS (qualidade do serviço).

Esse cenário de complexidade crescente resultou em uma demanda de serviços de TI maior do que a oferta e ameaça "minar" ou mesmo "arruinar" os muitos benefícios que a TI oferece, pois nas atuais taxas de expansão não haverá pessoal habilitado ou capacitado suficiente para seu gerenciamento. A demanda por profissionais de TI capacitados deve crescer acima de 100% nos próximos anos, e em países como os EUA, as vagas não preenchidas já somam centenas de milhares.

Mesmo com a demanda por pessoal qualificado atendida, a complexidade cresce além da capacidade humana para gerenciá-la, pois gera sobreposições, dependências e iterações que necessitam de intervenções e respostas que estão além de nossas habilidades – por exemplo, analisar causas de falhas e

implementar melhorias podem gerar problemas com mais variáveis do que um ser humano pode gerenciar.

É necessária uma nova abordagem para solucionar ou minimizar o problema da complexidade.

Clientes de TI querem ditar políticas de negócio para o sistema implementar os detalhes - se a tecnologia se tornar mais simples e fácil, aplicações novas e imprevisíveis irão surgir e mais pessoas irão utilizá-la.

Nesse sentido, a barreira da "inteligência das máquinas" não ameaça o progresso - na evolução da humanidade a automação tem tido papel fundamental.

Por exemplo, na agricultura, cerca de dois séculos de inovações em automatização de tarefas manuais tem trazido eficiências na produção e colheita. Nesse período a lavoura como percentual da força de trabalho diminuiu de 90% para 2,6%, e as horas de trabalho para produzir 100 alqueires de trigo caíram de 300 horas para 3 horas (fonte: *U. S. Department of Agriculture*).

Na telefonia, a implementação de um protocolo de software pela AT&T/ Bell na década de 20 permitiu o atendimento da demanda por telefones sem a necessidade de alocar mais operadores.

Muito pode ser feito para automatizar o funcionamento diário de sistemas computacionais sem engendrar pelos caminhos da inteligência artificial (AI), embutindo a complexidade na própria infra-estrutura de TI e automatizando seu gerenciamento, utilizando uma abordagem inspirada nos sistemas massivamente complexos do corpo humano.

O sistema nervoso autônomo dos seres humanos "cuida de tudo", regulando o batimento cardíaco, checando níveis de açúcar e oxigênio no sangue, monitorando a temperatura etc, sem qualquer reconhecimento consciente ou esforço de nossa parte. Ele controla todas as funções através de um grande número de condições externas, sempre mantendo o estado interno em equilíbrio (*homeostasis*) e preparando o corpo para as próximas tarefas.

Essa é a forma como é necessário construir sistemas computacionais - sistemas capazes de rodar a si mesmos, ajustando-se a várias circunstâncias e preparando seus recursos para manipular de forma mais eficiente as cargas incidentes sobre ele. Sistemas capazes de antecipar necessidades, permitindo a usuários que se concentrem em tarefas estritamente delegadas a seres humanos.

Essa é a abordagem proposta pela *Autonomic Computing* (AC).

Autonomic é um termo derivado da biologia que significa autônomo, independente, livre.

O termo "*Autonomic Computing*" (computação autônoma ou autonômica) ou simplesmente "AC", deriva do "*autonomic nervous system*" (sistema nervoso autônomo do ser humano), que controla suas funções vitais.
AC também é o termo utilizado pela IBM para descrever o conjunto de conceitos, tecnologias e ferramentas que habilitam aplicações, sistemas e redes a tornarem-se autogerenciáveis.

A *Autonomic Computing* (AC) visa o desenvolvimento de sistemas computacionais abertos e inteligentes, autogerenciados e auto-regulados, capazes de rodar com mínima intervenção humana, adaptando-se dinamicamente ao contexto conforme políticas e objetivos de negócio,

auxiliando na construção de infra-estruturas de TI mais automatizadas para reduzir custos, incrementar o *uptime* e fazer uso mais eficiente de recursos.

Apesar de a *Autonomic Computing* não depender da *Inteligência Artificial* (AI), esta deverá ajudar a alavancar a AC. Novas aplicações da teoria e leis do controle disponibilizam *insights* sobre como rodar sistemas complexos que otimizam seus ambientes. Porém, a AC não requer a duplicação da consciência humana (pensamento) como objetivo final. A *Autonomic Computing* não envolve poder cognitivo humano, mas sistemas que podem adaptar-se, aprender e assumir funções previamente executadas por pessoas, através de funcionalidades autônomas. Sistemas verdadeiramente autônomos ainda estão muito distantes.

Além da inteligência artificial e teoria do controle, outros campos de estudo científico que podem contribuir com a AC são:

- Sistemas adaptáveis complexos
- Teoria do caos
- Cibernética
- Sistemas auto-evolutivos (automonitoramento e auto-ajuste)
- Chips celulares capazes de se recuperar de falhas
- Gerenciamento de cargas heterogêneas
- Teoria de controle aplicada a ciência da computação

Como no corpo humano, a AC possui uma "hierarquia de auto-governança", onde cada nível mantém uma medida de independência enquanto contribui para um nível mais alto de organização.

Cada nível permanece "desatento" em relação às outras partes, tomando conta de si mesmo e pedindo ajuda a um nível mais alto quando necessário. Os

componentes contribuem para o funcionamento do sistema como um todo, sem interferências regulares.

Um sistema computacional autônomo pode ser descrito por possuir ao menos oito elementos-chave ou características [Horn, 2001]:

a) Para ser autônomo, um sistema computacional precisa conhecer a si mesmo e possuir componentes que tenham identidade sistêmica

- um sistema autônomo deve possuir conhecimento detalhado de seus componentes, status atual, capacidade limite e todas as conexões com outros sistemas para governar a si mesmo;

- deve conhecer a extensão de seus próprios recursos, aquele que ele pode "emprestar" ou "alugar" e aqueles que podem ser compartilhados ou isolados;

- essa definição de sistema é simples para uma máquina ou centenas de máquinas em rede dentro de uma empresa – mas conectando essas máquinas a milhões de máquinas na internet, torna-as interdependentes e permite uma audiência global comunicar-se com os outros computadores através de celulares, TVs e dispositivos inteligentes;

- um sistema não pode monitorar o que ele não sabe que existe, ou controlar pontos específicos se seu domínio de controle permanece indefinido;

- para construir essa habilidade em sistemas computacionais, políticas claramente definidas e incorporadas em agentes de software adaptáveis devem gerenciar sua própria definição de sistema e sua interação com os sistemas a sua volta – esses sistemas devem ter a capacidade de

"fundirem-se" automaticamente com outros sistemas para formar novos sistemas, mesmo que temporariamente, e separarem-se em sistemas discretos quando conveniente.

b) Um sistema computacional autônomo precisa conFigurar-se e reconFigurar-se sob condições variáveis e não esperadas

- *setup* deve ocorrer automaticamente;
- administradores não conseguem gerenciar e aplicar centenas ou milhares de variáveis ao mesmo tempo;
- para habilitar a conFiguração automática um sistema deve criar múltiplas imagens do software crítico (como uma clonagem) e realocar seus recursos quando necessário;
- em um sistema distribuído globalmente este precisará alavancar suas múltiplas imagens e salvaguardar cópias para recuperar-se de falhas em parte de sua rede;
- sistemas adaptativos podem "aprender" as melhores conFigurações para encontrar melhores níveis de performance.

c) Um sistema computacional autônomo nunca "se contenta" com o *status quo* – ele sempre busca formas de otimizar-se

- deve monitorar suas partes constituintes e ajustar (*fine-tune*) os fluxos para obter os objetivos de sistema pré-determinados - como um maestro escuta sua orquestra e ajusta suas características dinamicamente e expressamente pra obter uma interpretação musical particular;

- esse esforço consciente de auto-otimização é a única forma que um sistema possui para satisfazer as complexas e às vezes conflitantes demandas de TI de um negócio, seus clientes, fornecedores e funcionários;

- auto-otimização deve ser também a chave para habilitar disponibilidade onipresente do *e-sourcing* ou a entrega de serviços computacionais similar aos *utilities*;

- auto-otimização demanda mecanismos de controle de *feed-back* avançados para monitorar suas medidas (métricas) e tomar as ações apropriadas;

- inovações em teoria de controle aplicada à computação precisam ocorrer em paralelo com novas abordagens para arquitetura de sistemas geral, produzindo sistemas desenhados com objetivos de controle em mente;

- algoritmos buscando fazer decisões de controle precisam ter acesso as medidas internas e como os botões de controle em um rádio, os pontos de controle precisam afetar a fonte daquelas medidas internas;

- todos os componentes de um sistema computacional autônomo precisam ser controláveis de forma UNIFICADA.

d) Um sistema computacional autônomo precisa executar algo similar a um "curativo" (reparo) – precisa recuperar-se de eventos rotineiros e extraordinários que podem causar mau funcionamento de suas partes

- precisa estar preparado para descobrir problemas ou problemas potenciais e encontrar uma forma alternativa de utilizar recursos ou reconFigurar o sistema para manter seu funcionamento;

- ao invés de peças de substituição crescentes (células), o reparo em um sistema significa "chamar" elementos redundantes ou sub-utilizados para agir como peças sobressalentes;

- checagem de erro e correção (redes) e RAID são tipos de "reparo" utilizados pela computação nos últimos 50 anos;

- a complexidade da TI atual torna mais difícil localizar as causas dos problemas, até em ambientes mais simples, como em um *desktop* (solução freqüente: *shutdown / reboot*);

- em sistemas mais complexos identificar as causas de falhas pede uma análise de causa e efeito;

- respostas de reparo tomadas por sistemas computacionais autônomos devem seguir regras geradas por especialistas humanos – mas quando embutirmos mais inteligência eles começarão a descobrir novas regras por eles próprios, que ajudarão a utilizar redundância ou recursos adicionais para recuperar-se e encontrar o objetivo principal: satisfazer os objetivos especificados pelo usuário.

e) **Um mundo virtual é não menos perigoso que um físico, então um sistema computacional autônomo precisa ser um perito em auto-defesa**

- precisa detectar, identificar e proteger-se contra vários tipos de ataques para manter a segurança e integridade geral do sistema;

- ataques podem vir de qualquer lugar e vírus espalham-se rapidamente;

- mais do que responder a falhas de componentes ou rodar checagens periódicas por sintomas, um sistema autônomo deve manter-se alerta, antecipar ameaças e tomar ações necessárias – as respostas precisam endereçar dois tipos de ataques: vírus e intrusões;

- similar ao sistema imunológico humano, um "sistema imunológico digital" – uma abordagem atual pode detectar código suspeito, automaticamente enviá-lo para um centro de análise e distribuir uma "vacina" para o sistema – todo processo acontece sem o conhecimento do usuário;

- para lidar com ataques, sistemas de intrusão precisam automaticamente detectar e alertar os administradores de sistema para os ataques – normalmente peritos em segurança examinam o problema, analisam e reparam o sistema – como a escala de redes e sistemas continua expandindo-se e a probabilidade de ataques aumenta, será necessário automatizar o processo, pois não haverá peritos suficientes para tratar cada incidente.

f) Um sistema computacional autônomo conhece seu ambiente e o contexto envolvido em sua atividade, e atua em conformidade

- um sistema computacional autônomo deve encontrar e gerar regras para melhor interagir com sistemas vizinhos – deve ligar recursos recursos disponíveis, negociar elementos sub-utilizados com outros sistemas, alterar seu ambiente e a si mesmo no processo, enfim, ADAPTAR-SE;

- essa "sensibilidade-contextual" inclui melhorar os serviços baseado no conhecimento sobre o contexto da transação;

- provimento de informação útil ao invés de dados confusos – por exemplo, entregar todos os dados necessários para mostrar uma página web sofisticada obviamente sobrecarregaria um usuário conectado com um celular (tela pequena, menor capacidade de processamento etc) querendo apenas o endereço do banco mais próximo;

- sistemas computacionais autônomos devem ser capazes de descrever a si mesmos e seus recursos disponíveis para outros sistemas e serem aptos a automaticamente descobrir outros serviços no ambiente;

- compartilhamento de recursos de super-computadores via *grid* deve contribuir para as tecnologias necessárias para a habilidade de computação autônoma de "ambiente-atento";

- avanços são necessários para tornar os sistemas atentos para ações do usuário, acompanhado de algoritmos que permitem esse sistema determinar a melhor resposta em cada contexto.

g) **Um sistema computacional autônomo não pode existir em um ambiente fechado (hermético)**

- enquanto independente em sua habilidade de gerenciar a si mesmo, um sistema computacional autônomo precisa funcionar em um ambiente/mundo heterogêneo e implementar padrões abertos – não pode ser uma solução proprietária e fechada;

- na natureza, todos os tipos de organismos precisam coexistir e dependem um dos outros para sobreviver (a biodiversidade atua/ajuda a estabilizar o ecossistema) – nos sistemas computacionais a coexistência e interdependência análoga é inevitável – negócios conectam-se a fornecedores, clientes e parceiros, pessoas conectam-se a seus bancos, agentes de viagem e lojas favoritas – independentemente do hardware ou aplicações que estão usando;

- colaborações atuais na ciência da computação para criação de padrões abertos devem permitir novos tipos de compartilhamento;

- avanços em sistemas computacionais autônomos devem fundamentar-se em padrões abertos – formas padrão de identificação de sistemas de comunicação e negociação – talvez até novas classes de intermediários "*system-neutral*" ou agentes especificamente designados como "cyber-diplomatas" para regular demandas de recursos conflitantes – precisam ser inventados e acordados.

h) **Talvez mais crítico para o usuário, um sistema computacional autônomo deve antecipar os recursos otimizados necessários enquanto mantendo sua complexidade escondida**

- esse é o objetivo máximo da computação autônoma – gerenciar os recursos de TI para diminuir a distância entre o negócio ou objetivos individuais dos clientes, e a implementação de TI necessária para atingir esses objetivos – sem envolver o usuário nessa implementação;

- um sistema computacional autônomo deve permitir antecipação e suporte (como no sistema nervoso humano), fornecendo informações essenciais com um sistema otimizado e pronto para implementar as decisões que os usuários executam e não necessariamente envolvê-los em resultados paliativos do sistema.

Esses oito elementos-chave de sistemas computacionais autônomos podem ser consolidados em quatro características de autogerenciamento descritas como *self*-CHOP (configure, heal, optimize, protect) na Tabela 2.1.

Tabela 2.1: Características self-CHOP de sistemas AC

Característica	**Descrição**
auto-conFiguração	adaptação a ambientes dinamicamente mutantes
auto-reparação	descobrir, diagnosticar e agir para prevenir interrupções
auto-otimização	ajustar recursos e balancear cargas para maximizar o uso de recursos
auto-proteção	antecipar, detectar, identificar e proteger contra ataques

A *Autonomic Computing* está focada em tarefas complexas de manutenção e gerenciamento, liberando a equipe de TI para tarefas que precisem de reflexão, pensamento, opinião, inovação, criatividade ou oportunidade.

Sistemas AC permitem a correção e identificação de problemas de forma antecipada em relação a uma equipe, ou seja, computadores identificam e reparam seus próprios problemas antes de serem descobertos.

Dessa forma, aumenta-se a produtividade enquanto minimiza-se a complexidade para usuários – projetando e construindo sistemas computacionais capazes de rodar sozinhos, ajustando-se a várias circunstâncias e preparando seus recursos para manipular de forma eficiente as cargas submetidas.

A AC basicamente proporciona:

- Redução da complexidade computacional para equipe de TI e usuários (a AC "embute" a complexidade);
- Diminuição da demanda por conhecimentos especializados de TI;
- Aumento da capacidade computacional;
- Implementação acelerada de novas características;
- Processo de decisão melhorado para sistemas e pessoas.

Soluções AC possuem uma abordagem holística, tendo impactos na concepção, projeto, gerenciamento e manutenção da tecnologia – dessa forma, as empresas precisam estar preparadas para evoluir quase todos os aspectos

sobre como fazem negócios, traduzindo principalmente políticas de negócios em políticas de TI.

AC não é revolução, mas evolução gradual com tecnologias adotadas e implementadas passo a passo e em níveis - não se deve ir do conceito à implementação em um único passo.

Esse processo evolucionário está focado na melhoria das características de um sistema na direção de um estado ideal em um ambiente computacional.

A migração para ambientes de computação autônoma autogerenciados está baseada em um modelo de implementação, descrito na Tabela 2.2.

Tabela 2.2: Modelo de implementação AC

Nível	Descrição
(1) Básico	computação manual, dependente da equipe de TI
(2) Gerenciado	TI baseada em consoles, há redução do tempo de coleta e sintetização de informações
(3) Preditivo	reconhecimento de padrões e recomendação de ações
(4) Adaptável	ações baseadas em informações; equipe gerencia performance e níveis de serviço
(5) Autônomo	sistemas dinamicamente gerenciados por políticas e regras de negócio

Produtos, sistemas e ambientes podem ser classificados em um dos cinco níveis de maturidade, que mostram como uma empresa evolui em sua adoção da AC e nos processos e conhecimentos de suporte.

Atualmente, alguns ambientes de TI já se encontram no nível gerenciado ou preditivo.

Para implementar AC, apenas automatizar partes de sistemas computacionais não é suficiente. É uma operação de auto-gestão do sistema inteiro, e não só parte dele, que traz benefícios. Esse é o motivo da necessidade de uma abordagem sistêmica na construção de sistemas AC – uma visão holística que permitirá a entrega de muito mais automação do que a soma de suas partes autogerenciadas individualmente.

Ou seja, é necessário derivar um conjunto de regras procedimentais/comportamentais e iterativas embutidas em elementos autônomos individuais, que quando combinados produzem um comportamento global. Vide teoria da robusteza da AC, teoria da negociação e métodos para automação da agregação de variáveis estatísticas.

Para desenvolvimento de sistemas AC é necessário o progresso de duas frentes:

- fazer componentes de sistema individuais autônomos
- obter comportamento autônomo no nível de sistema de TI corporativo global

Para isso é necessário que seja feita uma avaliação de adequação a proposta AC [Murch, 2004], analisando dentre outros aspectos:

- se o sistema é o candidato "certo" para implementação da AC;
- se o sistema-alvo pode manipular uma aplicação autônoma;
- se essa aplicação é uma candidata para os conceitos autônomos;
- o nível de maturidade autônoma do negócio;
- as seis áreas funcionais (ambientais) da TI:

    - gerenciamento de segurança
    - provisionamento de usuários/recursos
    - gerenciamento de capacidade/performance
    - implementação de soluções
    - disponibilidade
    - gerenciamento de problemas

Para atingir o objetivo da AC é necessário que cada componente possa compartilhar informação com cada outra parte e contribuir para prontidão e regulação do sistema como um todo.

Esse objetivo pode ser alcançado através de uma arquitetura AC que incorpore interfaces e pontos consistentes de controle em ambientes heterogêneos.

Um sistema computacional autônomo pode ser dividido em partes ou camadas – essas partes são organizadas por uma arquitetura e conectadas através de uma infra-estrutura distribuída que permite que seus componentes colaborem utilizando mecanismos padrão, como *web-services*.

A arquitetura AC define um conjunto de blocos de construção cuja finalidade é [Draper, 2006]:

- prover os elementos básicos nos quais um sistema de gerenciamento autônomo pode ser construído

- definir como esses elementos devem interagir para satisfazer a função autônoma

- identificar as interfaces e modelos padrão que permitirão soluções integradas

Essa arquitetura pode ser imaginada como um barramento de serviços que integram os vários componentes, como (1) recursos gerenciados, (2) *touchpoints*, (3) *autonomic managers* e (4) consoles.

Recursos gerenciados são qualquer tipo de recurso, hardware ou software, que podem ter embutidos atributos autogerenciados. Formam a camada mais baixa do sistema.

Touchpoints (pontos de contato) são as interfaces para uma instância de um recurso gerenciado, como um sistema operacional ou servidor. Um *touchpoint* implementa "sensores" (*sensors*) que recebem e respondem a estímulos, e "executores" (*effectors*) que executam uma ação em resposta a um estímulo.

Autonomic managers (gerenciadores autônomos) automatizam alguma parte do processo de TI, incluindo orquestração de outros gerenciadores, que proporciona a característica de computação autônoma por todo o sistema. O *autonomic manager* é a estrutura que contém um *loop* de controle que manipula eventos.

A peça chave da AC é o *Autonomic Manager* (Figura 1), ele é responsável pela automatização das funções de gerenciamento de TI que tornam o sistema autogerenciado. Suas quatro funções primárias são [Draper, 2006]:

- Monitorar
- Analisar
- Planejar
- Executar

Autonomic Managers utilizam políticas (metas ou objetivos) que orientam como as funções primárias são executadas.

Segundo [Murch, 2004], as principais características que definem um *autonomic manager* são:

- Determinação de políticas
- Conhecimento sobre a solução
- Administração comum do sistema
- Determinação do problema
- Monitoramento autônomo
- Análise de complexidade
- Medição transacional

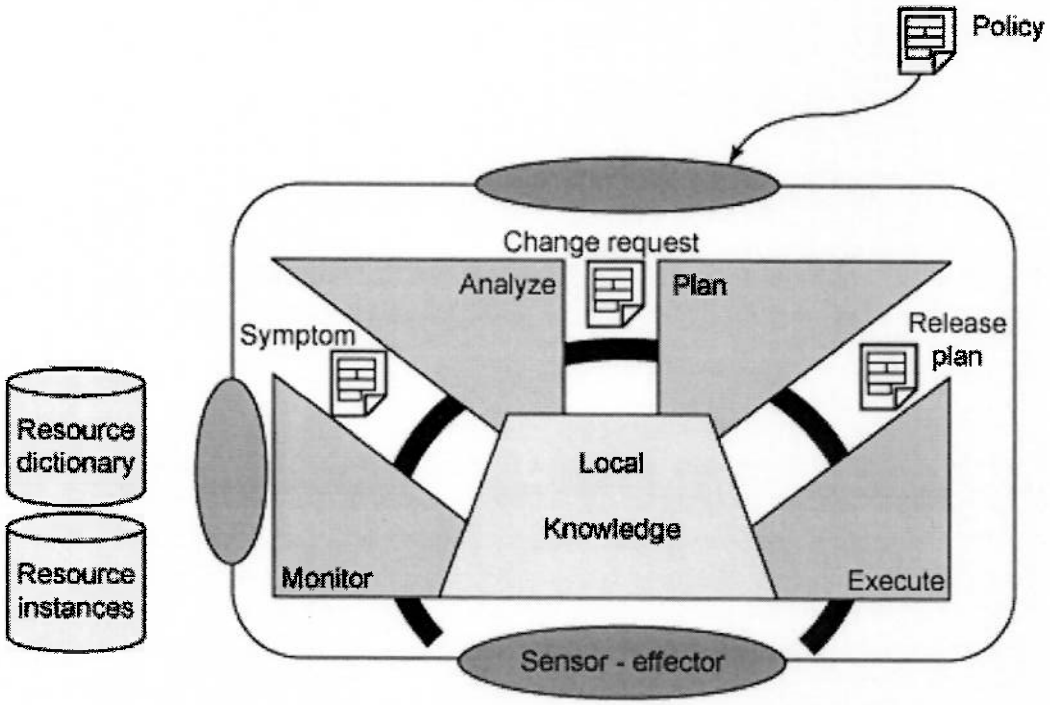

Figura 1: Representação de um autonomic manager

Um recurso pode ter um ou mais *touchpoint autonomic managers*, cada um implementando um loop de controle.

Consoles de soluções integradas são a interface de gerenciamento de sistema comum para a equipe de TI.

O ciclo de vida de um elemento autônomo começa quando o mesmo é projetado.

Como os elementos autônomos consomem e provêem serviços é necessário ter em mente em todas as fases de desenvolvimento que o mesmo precisa representar características, necessidades e preferências.

Alguns exemplos de aplicação prática da AC em sistemas:

- redirecionamento de cargas para acomodação de *jobs* prioritários;
- realocação de servidores e procedimentos sob demanda;
- reconhecimento de padrões que representam fraquezas e/ou ameaças e aplicação de reparos potenciais;
- determinação de hierarquias de segurança com conFiguração e implementação dos protocolos necessários;
- acesso a informações de múltiplos periféricos através de interfaces consistentes;
- impactos da AC nas diversas indústrias [ganek];
- hot swapping de código [appavoo];
- autonomic personal computing (CIM databases) [bantz];
- segurança em ambiente AC [chess];
- managing web server performance with autonomic agents [diao];
- network processors [haas];
- reconFiguração dinâmica de servidores [jann];
- IHC x AC [maglio];
- AC query optimizer para DB2 [markl];
- autonomous machines [norman];
- tuning [russel];
- mirroring de sites [verma];
- AC e proactive computing, novos dominios aplicação AC [want];
- modelo para reconFiguração dinâmica de sw [whisnant];
- algoritmos para seleção de componentes [yellin].

2.3 SOA

Por várias décadas projetistas desenvolveram sistemas baseados exclusivamente em requisitos técnicos - mas a mesma especificação de requisitos pode gerar sistemas diferentes - ou seja, requisitos tornam explícitas apenas algumas das propriedades do sistema final.

A especificação de requisitos é apenas o início – a falha em satisfazer outras restrições e necessidades pode levar a um sistema problemático e incompleto.

Ter um sistema aceitável envolve propriedades como performance, confiabilidade, disponibilidade, compatibilidade, utilização de recursos, segurança, alterabilidade, usabilidade e interoperabilidade com outros sistemas.

A arquitetura de *software* surgiu como parte crucial do processo de *design*, incorporando a estrutura de todos os tipos de sistemas.

Arquitetura de *software* de um programa ou sistema é a estrutura ou estruturas do sistema, que compreende os elementos de *software*, as propriedades visíveis externamente desses elementos e as relações entre eles [Bass, Clements, Kazman, 2003].

A arquitetura funciona como uma ferramenta importante de comunicação, discussão, análise e desenvolvimento de sistemas. Ela é resultado de influências técnicas, de negócio e sociais (ambiente) e ao mesmo tempo influencia este ambiente, que por conseqüência interfere em arquiteturas futuras. Esse ciclo é chamado de ciclo de negócio da arquitetura (ciclo *ABC*).

A visão arquitetural de um sistema é abstrata, não contendo detalhes de implementação, algoritmos ou representações de dados, concentrando-se sobre os procedimentos e interações de elementos "caixa-preta".

A arquitetura de software é o primeiro grande passo do *design* de um sistema que possui um conjunto de características desejadas.

As atividades envolvidas no gerenciamento da arquitetura de software são [Bass, Clements, Kazman, 2003]:

- criação do *business case*
- análise dos requisitos
- criação ou seleção da arquitetura
- documentação e comunicação da arquitetura
- análise ou avaliação da arquitetura
- implementação do sistema baseado na arquitetura
- certificação de conformidade da implementação

Não existe boa ou má arquitetura – arquiteturas podem ser mais ou menos adequadas a determinado propósito.

A SOA (*Service-Oriented Architecture* ou, em português, arquitetura orientada a serviços) é uma estratégia que visa orientar a construção dos ativos de software de uma empresa via metodologia de desenvolvimento orientada a serviços. Ela visa alinhar os objetivos de negócio com a área de TI, gerando um relacionamento sinérgico que visa agregar mais valor e gerar melhores resultados para o negócio.

SOA é também uma abordagem arquitetural de TI focada no negócio, que gera integração através da conexão de tarefas ou serviços, que são componentes de software construídos para que sejam facilmente integráveis com outros componentes de software e perfeitamente inteligíveis para as áreas de negócios, de forma a serem compartilhados e reutilizados por toda a empresa.

Segundo [Bass, Clements, Kazman, 2003], estilo ou padrão arquitetural é uma descrição de elementos e tipos de relacionamentos agrupados e com delimitações de uso.

Dessa forma, a SOA auxilia na construção de aplicações combinadas, que podem disponibilizar funcionalidades através de múltiplas fontes dentro e fora da empresa para suportar seus processos de negócio.

SOA incorpora um estilo de arquitetura corporativa (EA) que consolida um conjunto de serviços compostos para formação de outros serviços. Ela estabelece princípios de acoplamento livre, modularidade, encapsulação, reuso e componentização que produzirão a flexibilidade necessária para garantir aderência quanto a taxa de mudanças demandadas pelo negócio e alcançar produtividade, rentabilidade e competitividade melhoradas.

A SOA pode ter grande impacto positivo sobre estruturas grandes e/ou complexas. Para avaliar seu potencial de implementação, devem ser observadas as vantagens estratégicas, como arquitetura global de negócio (alinhamento) e as vantagens táticas do desenvolvimento orientado a serviços, como o potencial de reutilização de software e os ganhos de produtividade e agilidade.

A SOA parte do princípio que todos os negócios tem um *business design*, que descreve como o negócio funciona, que processos executa, sua estrutura organizacional, suas metas e objetivos de curto e longo prazo.

Derivando o design dos sistemas de TI do *business design* possibilita maior flexibilidade quanto a mudanças no negócio.

Essa relação é representada pelo "ciclo de vida SOA" (Figura 2), no qual a solução é implementada nos quatro estágios descritos na Tabela 2.3.

Tabela 2.3: Etapas do ciclo de vida de implementação SOA	
Etapa	**Descrição**
(1) Modelagem	• Elicitação de requisitos • Modelagem e simulação • Design
(2) Construção	• Exploração • Construção e testes • Integração
(3) Implementação	• Integração de pessoas e processos • Gerenciamento e integração da informação
(4) Gerenciamento	• Gerenciamento de aplicações e serviços • Gerenciamento de identidades e conformidades • Monitoramento de métricas de negócio

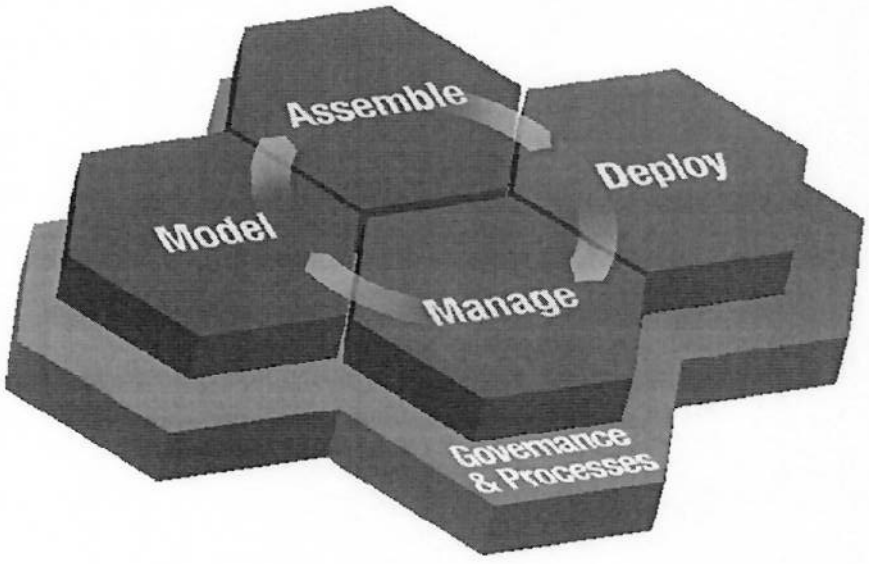

Figura 2: Ilustração do ciclo de vida SOA [IBM, 2005]

O ciclo de vida SOA faz parte da "IBM SOA *Foundation Architecture*", ou seja, o alicerce da arquitetura SOA.

Fazem parte também dos fundamentos da arquitetura SOA, um modelo lógico de arquitetura, seus elementos de suporte, um modelo de programação e um modelo físico arquitetural.

A SOA Solution Stack é um modelo em camadas para suporte na implementação de soluções (Figura 3).

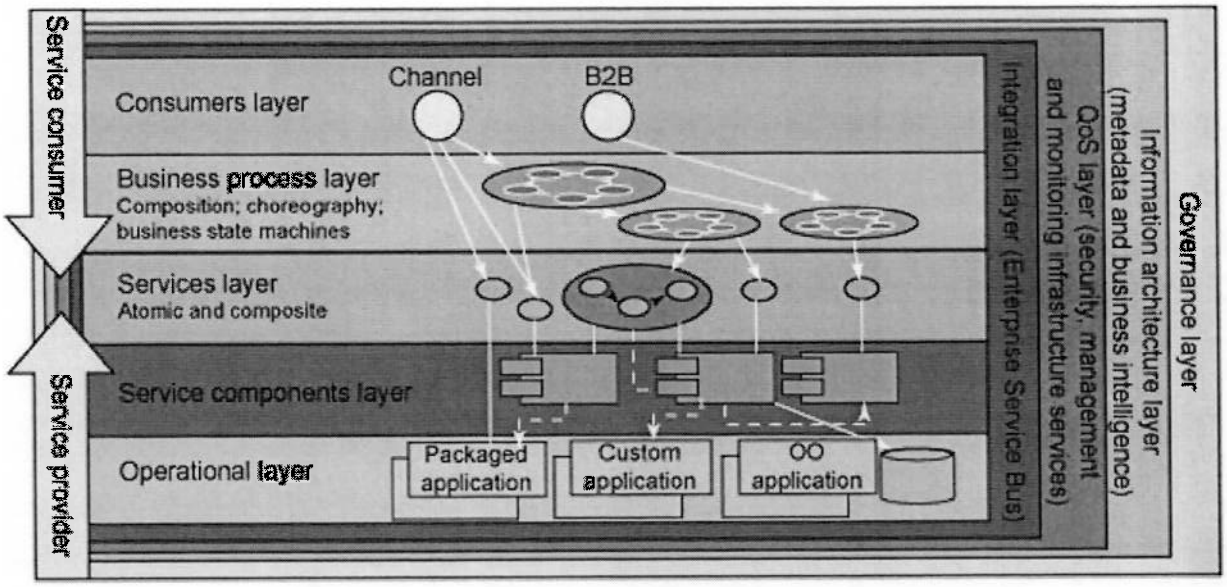

Figura 3: SOA Solution Stack [IBM, 2005]

O modelo lógico de arquitetura (Figura 4) decompõe a camada de serviços da aplicação SOA (*middleware*).

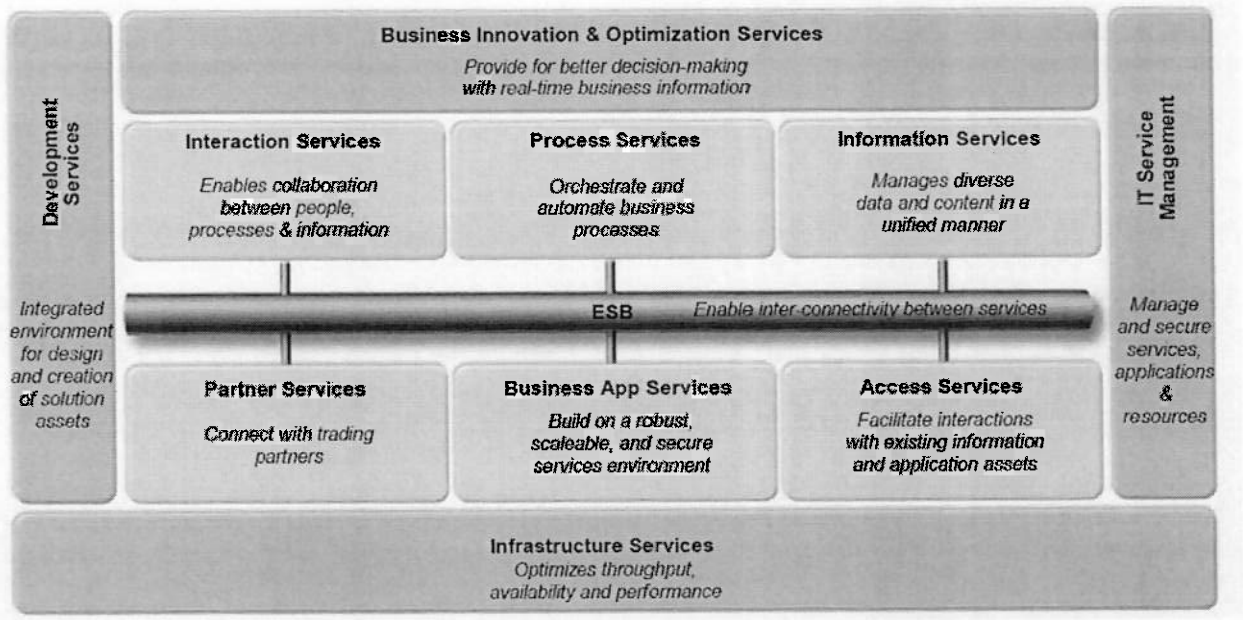

Figura 4: Middleware SOA [IBM, 2005]

2.4 SOA e AC como solução para micropagamentos

A proposta deste trabalho é aplicar SOA e AC em conjunto visando à geração de uma plataforma de micropagamentos flexível, autogerenciada, escalável e de baixo custo.

A ênfase na arquitetura deve-se principalmente por sua direta conexão com muitos riscos de projeto, principalmente no desenvolvimento da primeira geração de uma aplicação. Ela inclui os mais importantes blocos de construção de um sistema de software e suas interfaces. A arquitetura disponibiliza um esboço do sistema, compreendendo talvez de 10% a 20% da quantidade final de código [Kroll, Kruchten, 2003].

A arquitetura também pode ser definida como o conjunto de decisões acerca dos seguintes itens [Booch, Rumbaugh, Jacobson, 2000]:

- A organização do sistema de software;
- A seleção dos elementos estruturais e suas interfaces, que compõem o sistema;
- Seu comportamento, conforme especificado nas colaborações entre esses elementos;
- A composição desses elementos estruturais e comportamentos em subsistemas progressivamente maiores;
- O estilo de arquitetura que orienta a organização: os elementos estáticos e dinâmicos e suas respectivas interfaces, colaborações e composições.

Arquitetura é mais do que o resultado de requisitos funcionais de um sistema – é o resultado do background do arquiteto, o ambiente técnico e os objetivos da organização;

Entre rascunhos e arquiteturas bem desenhadas, com toda a informação apropriada sobre o sistema existem muitos estágios – cada estágio representa um conjunto de decisões arquiteturais – o agrupamento de escolhas arquiteturais – alguns desses estágios arquiteturais são muito úteis.

SOA é um padrão ou estilo arquitetural, que descreve elementos e tipos de relacionamento (Ex.: C/S), possui soluções conhecidas para problemas de performance, disponibilidade, segurança etc e exibe atributos de qualidade conhecidos orientando escolhas não é aleatórias.

A *Autonomic Computing* suporta e complementa a SOA, habilitando uma infra-estrutura autogerenciada flexível que responde as necessidades de processos e serviços componentizáveis que instanciam arquiteturas orientadas a serviços.

Ela também suporta e habilita integração e gerenciamento de serviços de acordo com os princípios e práticas SOA.

A arquitetura AC é uma arquitetura orientada a serviços que define um conjunto de blocos de construção arquiteturais para geração de sistemas de gerenciamento AC.

SOA e AC são tecnologias complementares e a base técnica para soluções autogerenciadas.

AC suporta e complementa arquiteturas orientadas a serviço através da alavancagem de uma infra-estrutura de TI flexível e autogerenciada, necessária para criar valor no cenário de negócios atual.

Pela abordagem SOA, os serviços de TI são consumidos por usuários finais fora da área de TI, como também por linhas de negócio dentro da organização.

Serviços podem ser consumidos através de portais ou por aplicações de negócio.

Aqueles que são automatizados utilizando SOA são percebidos como componentes que implementam serviços e fluxos de processo dentro da área de TI. Dessa forma, coordenam-se atividades executadas pela equipe de TI e tarefas automatizadas executadas por ferramentas de gerenciamento.

Em alguns casos serviços não podem ser automatizados, devendo-se proceder com uma abordagem manual para executá-los.

Tarefas selecionadas dentro dos fluxos de processos e serviços de TI são mapeadas para monitorar, analisar, planejar ou executar funções do *autonomic manager*, que tipicamente relaciona-se com ferramentas de gerenciamento.

Isso supre o componente de serviço com tarefas automatizadas (Figura 5).

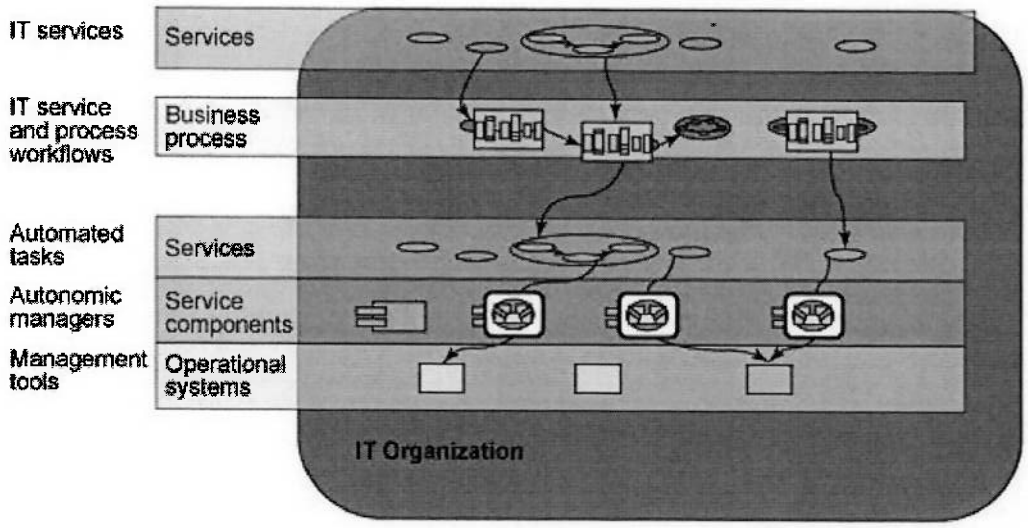

Figura 5: Tarefas automatizadas e autonomic managers [Draper, 2006]

A Figura 6 apresenta a automatização entre serviços internos/externos área de TI, utilizando a SOA Solution Stack.

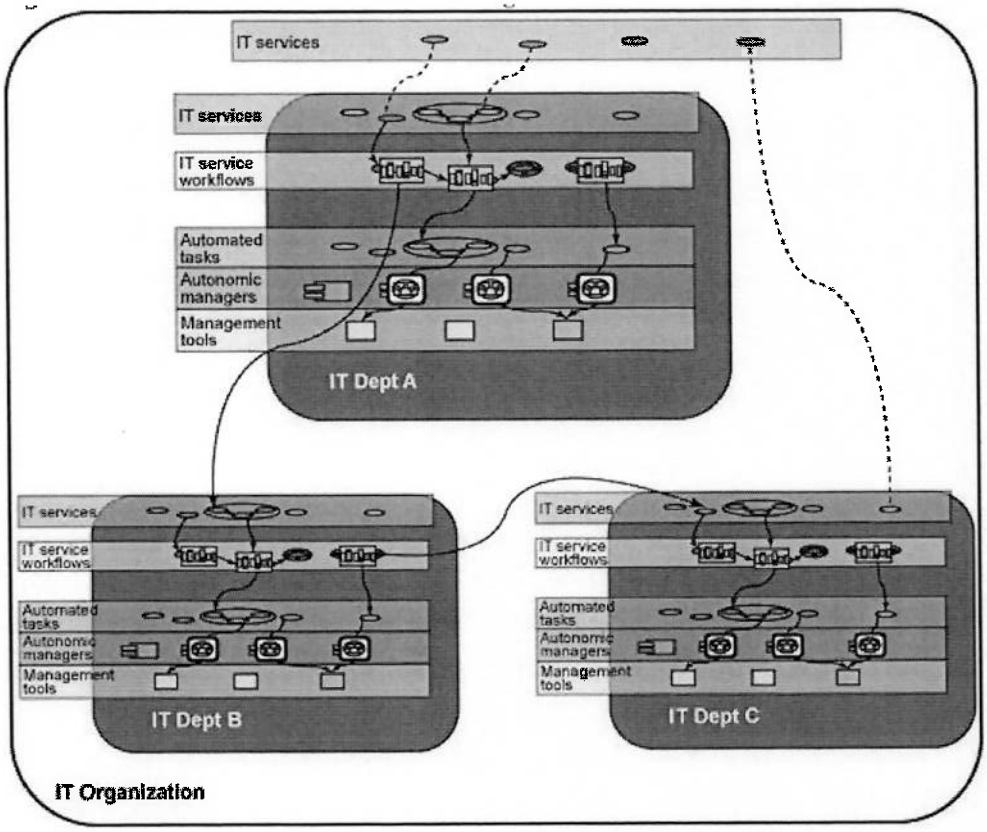

Figura 6: Serviços externos disparados por serviços internos a TI [Draper, 2006]

Baseado nestes conceitos de interoperabilidade arquitetural entre SOA e AC, o próximo capítulo apresenta a solução proposta de forma estruturada.

3 SOLUÇÃO PROPOSTA

A solução proposta para geração da arquitetura lógica do sistema em epígrafe prevê a utilização da metodologia SOMA [Arsanjani, 2004] para modelagem e arquitetura orientada a serviços. Ela é baseada no template SOA também chamado "*solution stack*", que apresenta um modelo para implementação de soluções SOA de nove camadas alinhadas aos processos de negócio. O resultado é a apresentação da arquitetura lógica final consolidada com o seguinte formato:

- Esboço arquitetural baseado na SOA Solution Stack
- Detalhamento do escopo arquitetural
- Camada de Consumo
- Camada de Processos de Negócio
    - Processos de negócio
    - Decisões arquiteturais
- Camada de Serviços
    - Portfólio categorizado de serviços (visão *middleware* em sub-camadas)
    - Decisões arquiteturais
- Camada de Componentes de Serviços
    - Áreas funcionais suportadas x componentes
    - Domínios de negócio, objetivos e processos suportados
    - Decisões relativas a governança
    - Decisões arquiteturais
- Camada Operacional

As camadas de Integração, Qualidade de Serviço, Arquitetura da Informação e Governança não serão consideradas devido ao foco do trabalho.

Face às características de projeto (projeto novo, não existem ativos legados) será utilizada a abordagem de implementação *top-down*, que é *business-driven* (decomposição de domínio) e a mais próxima aos fundamentos e ao ciclo de vida SOA (orientação negócio para TI).

A metodologia incorpora a identificação, especificação e realização de serviços, com base na descrição dos processos de negócio.

A incorporação de características AC na arquitetura SOA será realizada com base no modelo proposto por Christine Draper [Draper, 2006].

4 DESENVOLVIMENTO DA ARQUITETURA

O objetivo deste capítulo é desenvolver a base para geração da arquitetura lógica da solução.

4.1 Cenário proposto

Será utilizada metodologia SOMA [Arsanjani, 2004] para modelagem de processos de negócio e serviços, cuja realização (derivação) dará origem ao modelo de sistema (decomposição arquitetural do *business design* em um conjunto de componentes do sistema). Destaca-se como cenário proposto e foco do trabalho o desenvolvimento da arquitetura lógica com base em processos selecionados que particularmente demonstrem a aplicação da solução SOA e aplicação de SOA e AC em conjunto (não serão tratados aspectos de implementação física).

4.2 Modelagem do negócio

Esta seção descreve o contexto e aspectos chave do negócio para geração da arquitetura.

4.2.1 Contextualização

O negócio de micropagamentos tem como objetivo intermediar as transações financeiras no comércio de conteúdo digital de baixo valor unitário de forma fluída, rápida e segura para o comprador e rentável para o vendedor de conteúdo e para o próprio negócio. O custo unitário de processamento, pelos meios tradicionais de pagamento, tende a ser muito maior que a receita obtida e o volume de transações muito alto. Os valores cobrados são variáveis e definidos por produto ou serviço, pelo vendedor. Compradores adquirem créditos pré-pagos, consultam saldos e extratos

no portal do negócio e gastam seus créditos com acesso a conteúdo pago (fechado) diretamente nos canais de distribuição (*websites* na internet e portais móveis provedores de conteúdo). Vendedores disponibilizam seu conteúdo pago e acessam o portal do negócio para consulta a saldos e extratos de operação. Além de compradores e vendedores, outros envolvidos incluem funcionários operacionais e gerentes. As áreas funcionais do negócio são: tecnologia, atendimento, marketing, vendas, finanças e pessoal. Aspectos como segurança, disponibilidade, alterabilidade, performance, usabilidade e escalabilidade dos sistemas internos são fundamentais. As qualidades esperadas do negócio são *time to market*, custo/benefício, abrangência de mercado-alvo e integração com parceiros de negócio (vendedores de conteúdo). A Figura 7 apresenta o diagrama de contexto inicial.

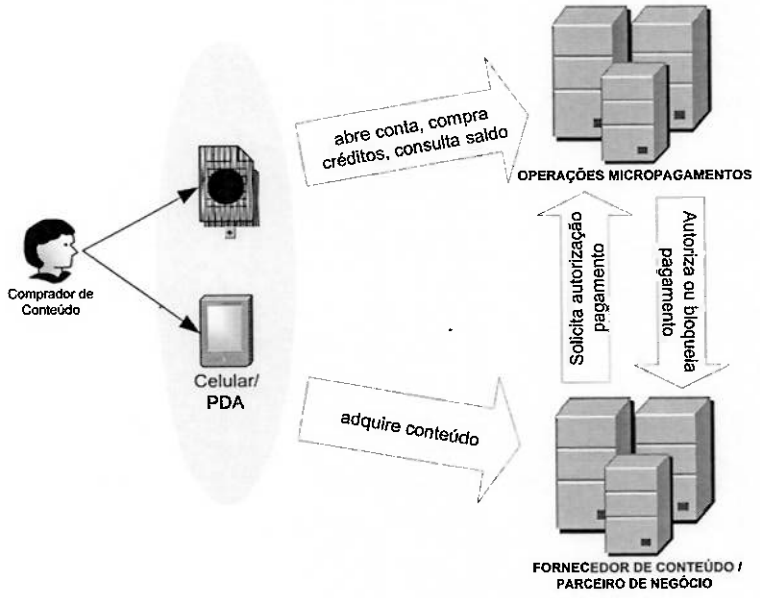

Figura 7: Diagrama de contexto inicial

4.2.2 *Escopo SOA / AC*

A implementação de uma solução SOA deve ter embasamento e compromisso estratégico, com escopo bem definido. Pode-se adotar uma estratégia global ou parcial, incremental ou não, com reutilização ou não de ativos legados, por exemplo. Para o foco deste trabalho (cenário proposto) será adotada uma estratégia parcial de implementação, compreendendo três processos chave do negócio, com as seguintes características:

- Exposição e consumo interno de pelo menos um processo de negócio como serviço;

- Exposição e consumo externo de pelo menos um processo de negócio como serviço;

- Exposição e consumo interno de pelo menos um processo de negócio como serviço implementando conceitos de *Autonomic Computing*.

4.2.3 Objetivos de negócio

Esta seção apresenta as questões chave de negócio e TI para adoção da solução SOA para o cenário apresentado.

As necessidades de negócio são:

- **[N-01]: atender a múltiplos canais de clientes (compradores) com máxima otimização de redundâncias**
  Para obter ganhos de escala é necessário criar uma plataforma que atenda a vários canais simultaneamente, por exemplo, web/internet e celulares;

- **[N-02]: otimizar ROI (*return on investment*) e TCO (*total cost of ownership*) da area de TI**
  Utilizar recursos técnicos para otimização de recursos humanos;

- **[N-03]: minimizar falhas e paradas nos processos relacionados com canais externos**
  Os processos de crédito (compradores de conteúdo) e autorização de pagamentos (fornecedores de conteúdo) devem garantir máxima confiabilidade e performance.

Os desafios de TI são:

- **[T-01]: alinhamento com o negócio**
  Garantir que estratégia e recursos de TI estejam alinhados com as premissas e estratégias de negócio.

A Figura 8 abaixo representa o contexto de negócio de alto nível proposto.

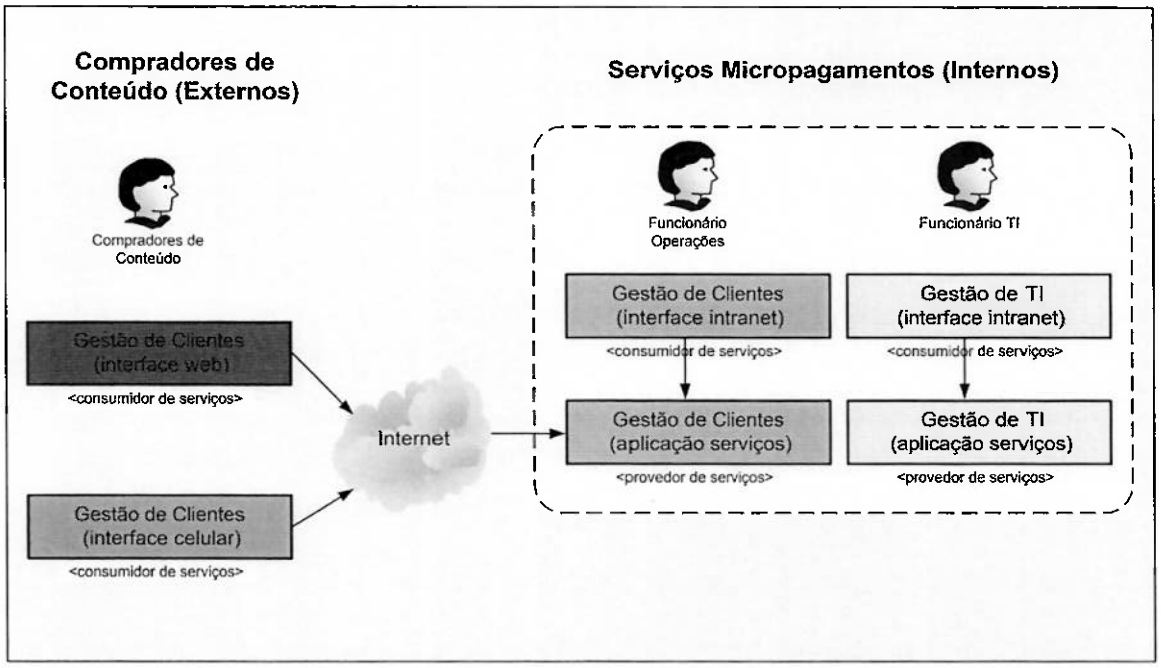

Figura 8: Representação do contexto de negócio de alto nível

4.3 Levantamento de Requisitos

Esta seção inclui os requisitos funcionais e não funcionais para o cenário proposto.

4.3.1 Requisitos funcionais

Os requisitos funcionais para o cenário proposto são descritos na Tabela 4.1.

Tabela 4.1: Requisitos Funcionais da Solução de Micropagamentos

Necessidade de Negócio	Requisito Funcional	Descrição
N-01	FR-01	Criar aplicação cliente interna baseada em web services para gestão de créditos de clientes (compradores de conteúdo)
	FR-02	Expor web services selecionados para clientes
	FR-03	Criar modelos baseados em web services para consumo dos serviços pelos canais web/internet e celulares
N-02	FR-04	Criar aplicação cliente interna baseada em web services para monitorar, analisar, planejar e executar ações sobre tentativas automatizadas não autorizadas de acesso externo e interno a serviços
	FR-05	Expor web services selecionados para a área de TI
	FR-06	Criar modelos baseados em web services para consumo dos serviços via canal intranet
N-03	FR-07	Criar aplicação cliente interna baseada em web services para monitorar, analisar, planejar e executar ações sobre recursos indisponíveis
	FR-08	Expor web services selecionados para a área de TI

Tabela 4.1: Requisitos Funcionais da Solução de Micropagamentos		
Necessidade de Negócio	**Requisito Funcional**	**Descrição**
	FR-09	Criar modelos baseados em web services para consumo dos serviços via canal intranet

4.3.2 Requisitos não funcionais

Os requisitos não funcionais para o cenário proposto são descritos na Tabela 4.2.

Tabela 4.2: Requisitos não funcionais da Solução de Micropagamentos		
Necessidade de Negócio	**Requisito Não-Funcional**	**Descrição**
N-01	NFR-01	Garantir acessibilidade, integridade e confidencialidade das informações para os canais disponibilizados
N-02	NFR-02	Garantir tratamento a tentativas de uso não autorizado de serviços e gerar ações corretivas e preventivas automatizadas sempre que possível
N-03	NFR-03	Monitorar performance e disponibilidade de serviços e gerar ações corretivas e preventivas automatizadas sempre que possível

4.3.3 *Diagrama de contexto*

A Figura 9 apresenta o diagrama de contexto atualizado como parte da solução proposta.

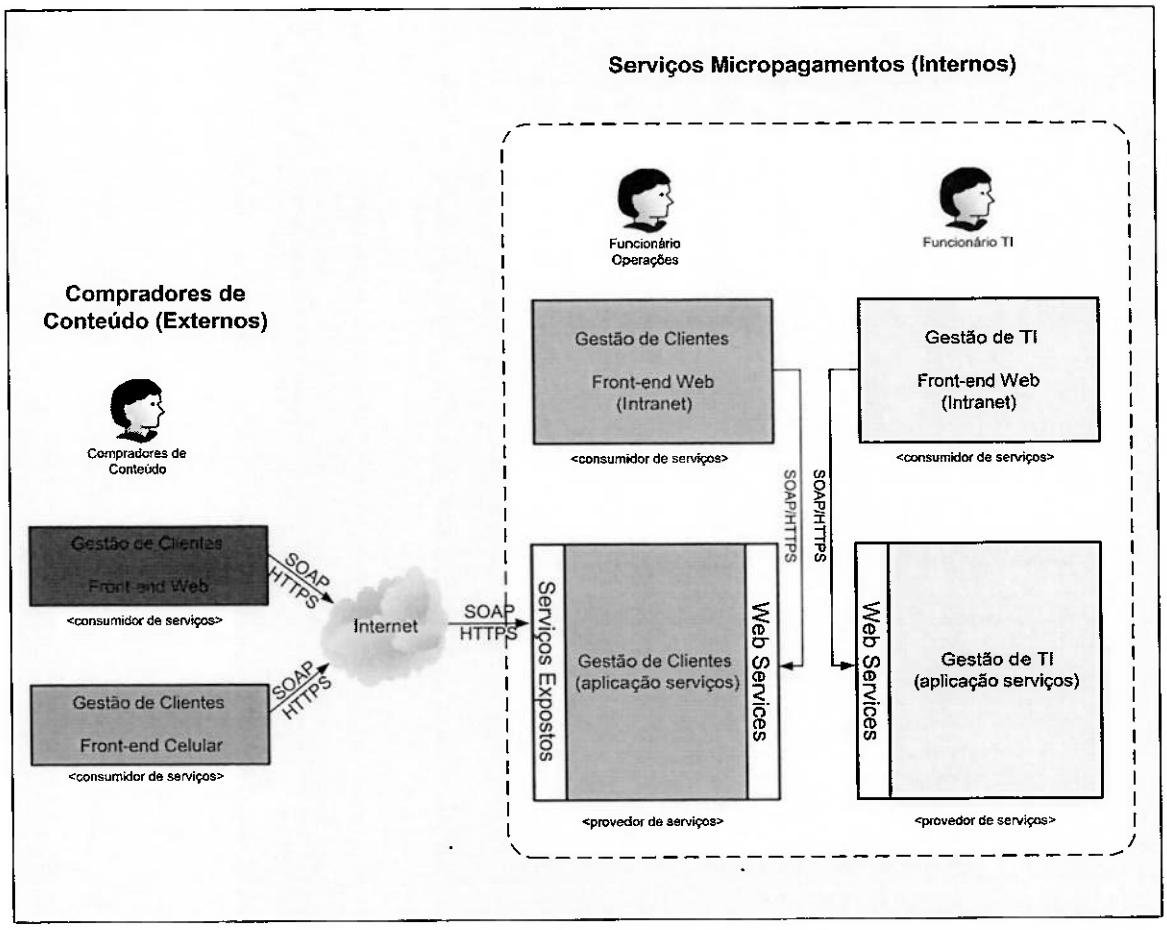

Figura 9: Diagrama de contexto de negócio atualizado

4.4 Identificação e design dos serviços

Métodos OOAD tradicionais não endereçam os três elementos chave da SOA: serviços, fluxos e componentes. Para uma solução SOA é necessário realizar decisões chave de arquitetura sobre cada camada da *SOA solution stack*. A metodologia SOMA [Arsanjani, 2004] disponibiliza técnicas para identificação, especificação e realização de serviços. Conforme discutido anteriormente, será adotada abordagem *top-down* para execução desta etapa.

4.4.1 Identificação de serviços

Para identificar a lista de candidatos a serviços em abordagem *top-down* deve ser realizada uma decomposição de domínio de negócio nas principais áreas funcionais e subsistemas, incluindo também em uma etapa adicional a decomposição de processos, sub-processos e casos de uso de alto nível, conforme apresentado na Figura 10.

Para o cenário apresentado, os domínios a serem focados são os de "gestão de clientes" (N-01) e "gestão de TI" (N-02 e N-03), com áreas funcionais "depósitos", "segurança" e "suporte", respectivamente.

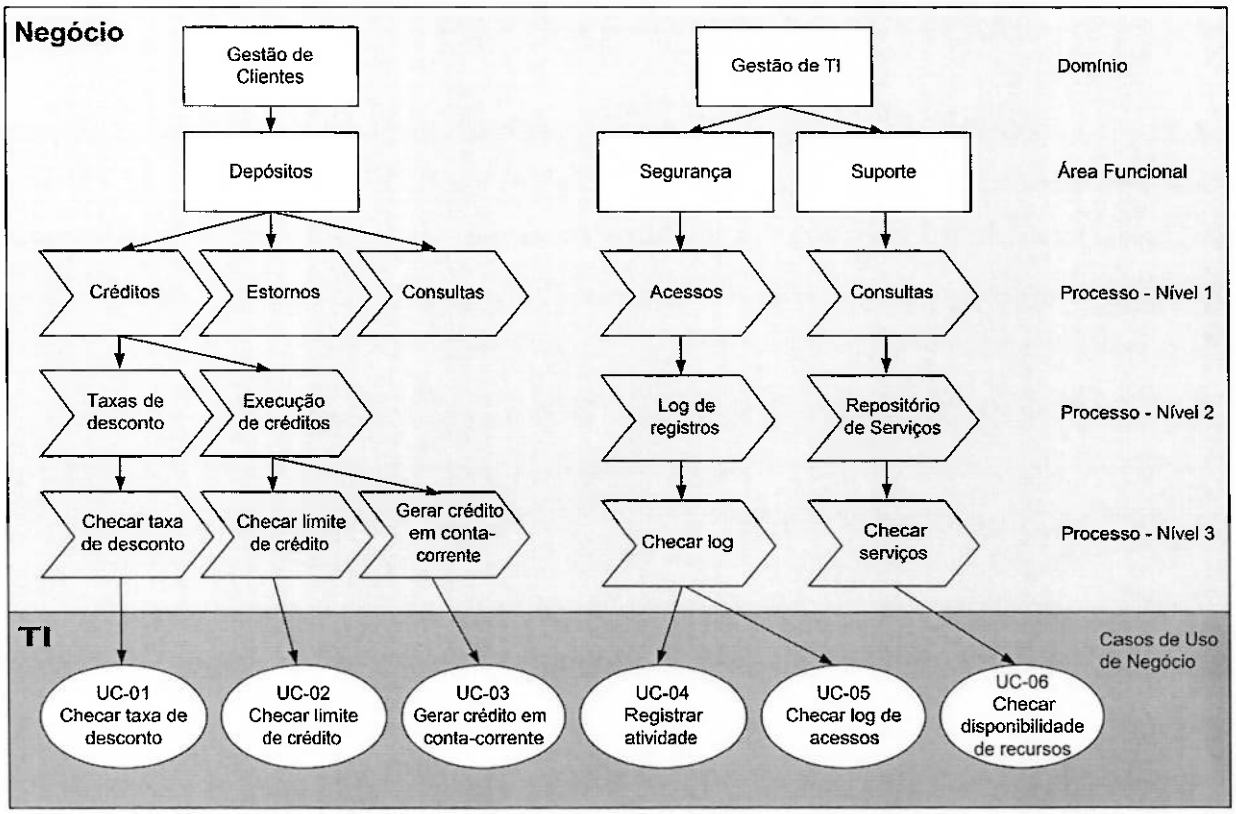

Figura 10: Decomposição de domínios da solução

A modelagem de casos de uso identificou os atores e casos de uso descritos nas Tabelas 4.3 e 4.4 a seguir:

Tabela 4.3: Identificação de atores da solução
Cliente (usuário final)
Operador Atendimento (suporte ao cliente)
Sistema

Tabela 4.4: Identificação de casos de uso da solução	
ID	**Caso de Uso**
UC-01	Checar taxas de desconto
UC-02	Checar limite de crédito
UC-03	Gerar crédito
UC-04	Registrar atividade
UC-05	Checar log de acessos
UC-06	Checar disponibilidade de recursos

Segundo a metodologia SOMA, casos de uso de alto nível geralmente são muito bons candidatos a serviços expostos para toda a organização ou aqueles utilizados nos limites da organização, entre *LOBs* (linhas de negócio).

A Tabela 4.5 contém a lista preliminar de serviços identificados com base nos casos de uso gerados na decomposição de domínios.

Tabela 4.5: Lista preliminar de serviços identificados na decomposição de domínio	
Serviço	**Descrição**
Checar taxas de desconto	Checa taxas de desconto a serem aplicadas sobre os depósitos / créditos em conta-corrente
Checar limite de crédito	Checa o limite para créditos em conta-corrente
Efetuar crédito	Gera crédito em conta-corrente proveniente de depósito realizado pelo cliente
Localizar crédito	Localiza créditos com base nos dados de seu registro
Estornar crédito	Estorna créditos realizados erroneamente
Registrar atividade	Gera registro da transação na base de logs
Checar log de acessos	Checa o log de acessos e adiciona endereços IP suspeitos a lista de bloqueios
Checar disponibilidade de recursos	Testa disponibilidade de recursos

A Figura 11 apresenta os resultados preliminares da implementação SOA na *Solution Stack* referentes aos serviços identificados na abordagem top-down.

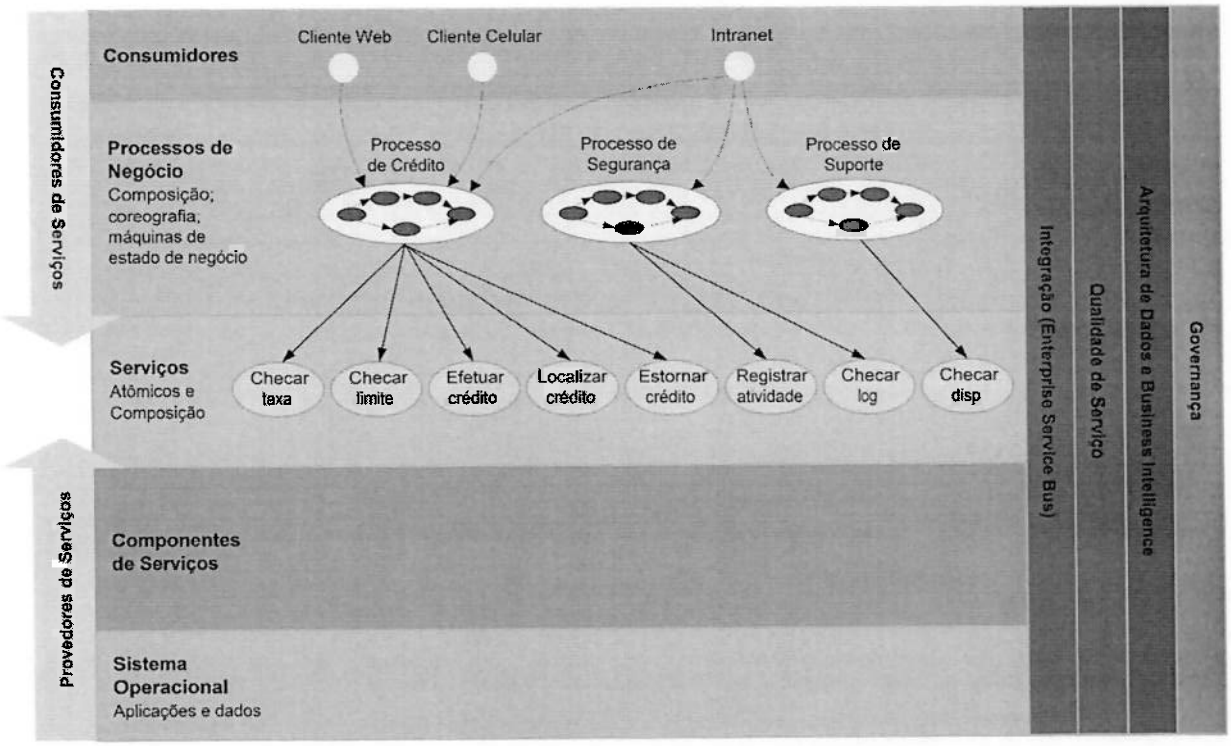

Figura 11: Resultados da implementação *top-down* projetadas nas camadas da SOA

Após a primeira etapa de decomposição de domínios, a metodologia SOMA sugere a aplicação do *Goal-to-Service Model*, uma abordagem "*meet-in-the-middle*" que permite garantir a rastreabilidade e alinhamento dos serviços identificados preliminarmente com os objetivos de negócio. Essa validação de completeza compara e analisa a lista de serviços identificados na primeira etapa contra ativos existentes e objetivos primários e secundários de negócio dentro do escopo de projeto, revelando novos candidatos a serviços.

Tabela 4.6: Implementação da abordagem "Goal-to-Service"			
Objetivo	**KPI**	**Métrica**	**Serviço**
Reduzir custos e perdas com tentativas de fraude e indisponibilidades	%	Registrar tentativas de acesso e indisponibilidades de serviços	Checa limite crédito Registra atividade Checa logs Checa disponibilidade

A avaliação do objetivo de negócio "Reduzir custos e perdas com tentativas de fraude e indisponibilidades" contra os serviços previamente identificados demonstra rastreabilidade e atendimento desta demanda. Para efeitos do cenário proposto não existem ativos pré-existentes para complemento analítico dessa etapa da metodologia (abordagem *bottom-up*).

Dessa forma, tanto o portfólio de serviços identificados quanto a projeção desses serviços nas camadas SOA da *solution stack* permanecem inalteradas para o cenário proposto.

Concluída a fase de identificação, a próxima etapa da metodologia SOMA é a categorização de serviços.

4.4.2 *Categorização de serviços*

A classificação ou categorização de serviços é iniciada após a identificação de serviços candidatos, em uma hierarquia que reflete a natureza composta ou fractal dos serviços. Serviços podem e devem ser compostos por componentes e serviços de granulação fina. A classificação auxilia na determinação de composição e camadas e é normalmente identificada baseada na análise de área funcional.

Tabela 4.7: Categorização de serviços da solução

Categoria	Serviço	Descrição
Crédito	Checar taxas de desconto	Checa taxas de desconto a serem aplicadas sobre os depósitos / créditos em conta-corrente
	Checar limite de crédito	Checa o limite para créditos em conta-corrente
	Efetuar crédito	Gera crédito em conta-corrente proveniente de depósito realizado pelo cliente
	Localizar crédito	Localiza créditos com base nos dados de seu registro
	Estornar crédito	Estorna créditos realizados erroneamente
Segurança	Registrar atividade	Gera registro da transação na base de logs
	Checar log de acessos	Checa o log de acessos e adiciona endereços IP suspeitos a lista de bloqueios
Suporte	Checar disponibilidade de recursos	Testa disponibilidade de recursos

4.4.3 *Especificação e realização de serviços*

As decisões sobre especificação e realização da especificação de serviços e componentes estão fora do escopo e do cenário proposto para este trabalho.

4.4.4 *SOA e Autonomic Managers como componentes de automatização*

Autonomic Managers são inseridos na arquitetura SOA como componentes de serviços para prover automatização.

A próxima seção ("Resultados") destaca sua inserção na arquitetura final para o cenário proposto.

A definição de políticas de orientação das funções de monitoramento, análise, planejamento e execução não faz parte do escopo proposto.

5 RESULTADOS E DISCUSSÃO

Esta seção apresenta os resultados da aplicação da SOA e de conceitos de Autonomic Computing para o negócio de micropagamentos.

5.1 Apresentação da arquitetura lógica para o cenário proposto

A Figura 12 mostra a representação lógica da arquitetura final para o cenário proposto com base na *SOA Solution Stack*.

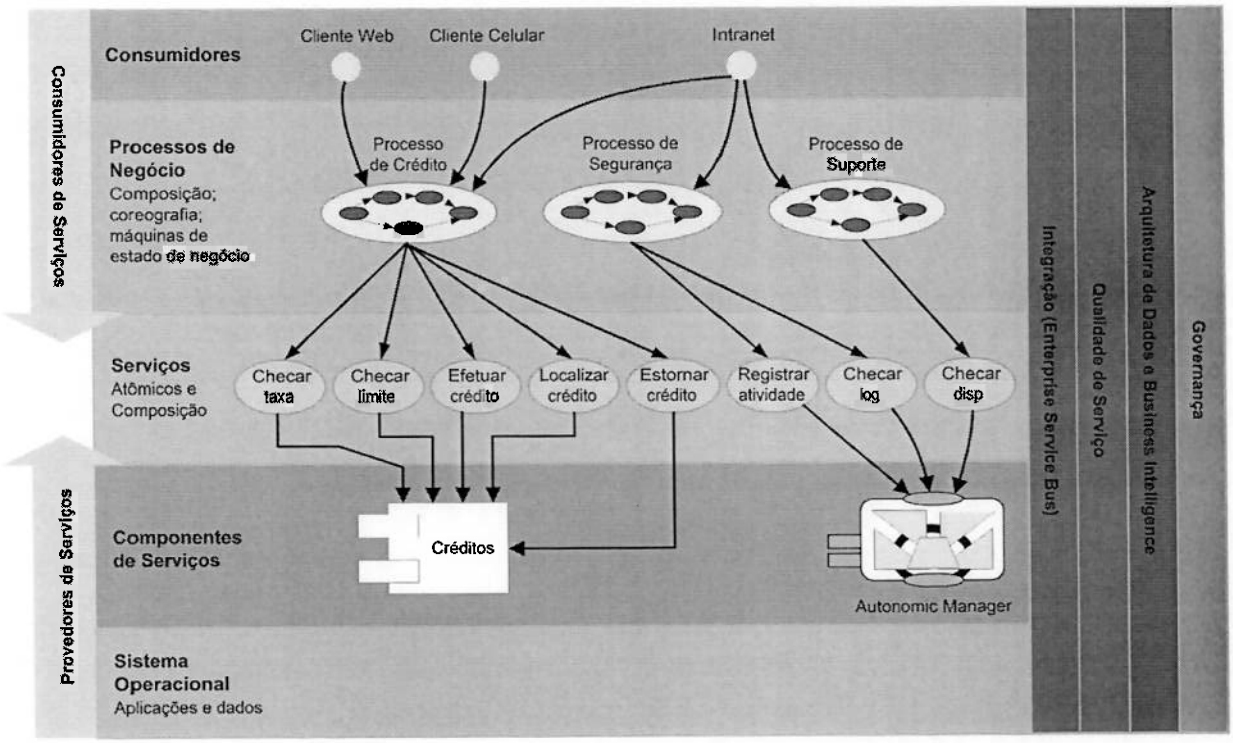

Figura 12: Resultado final da implementação na SOA Solution Stack

5.1.1 *Detalhamento do escopo arquitetural*

A representação arquitetural da Figura XXX incorpora a implementação dos três processos do cenário proposto, visando a rastreabilidade das implementações de TI com o negócio, a multiplicidade de canais e a otimização e automatização de recursos providos pelos conceitos de Autonomic Computing incorporados na arquitetura. Esta visão arquitetural pode ser estendida para outros processos de negócio utilizando-se a mesma metodologia.

5.1.2 *Camada de Consumo*

Esta camada proporciona a utilização de múltiplos canais como interface para os serviços disponibilizados pela plataforma. Foram pré-determinados dois canais para acesso de clientes (compradores de conteúdo) – web/internet e celular – que podem ser estendidos a outros canais, proporcionando reuso pela simples invocação das novas interfaces aos web services projetados. Um canal para acessibilidade de funcionários (intranet) também foi implementado na arquitetura.

5.1.3 *Camada de Processos de Negócio*

Esta camada apresenta os três processos de negócio selecionados para demonstração da arquitetura.

5.1.3.1. Processos de negócio

Os processos de negócio selecionados para o escopo proposto foram:

- Processo de Crédito
- Processo de Segurança
- Processo de Suporte

5.1.3.2. Decisões arquiteturais

Os processos de negócio foram definidos com base nas técnicas de modelagem de negócio descritos na metodologia SOMA. Não foram identificados processos candidatos a coreografias.

5.1.4 Camada de Serviços

Esta camada é composta por serviços identificados após a aplicação das técnicas de identificação de serviços e modelagem da metodologia SOMA. O portfólio categorizado de serviços identificados é listado a seguir.

5.1.4.1. Portfólio categorizado de serviços

A tabela 5.1 apresenta o portfólio categorizado de serviços identificados para o cenário proposto:

Tabela 5.1: Portfólio categorizado de serviços

Categoria	Serviço
Crédito	Checar taxas de desconto
	Checar limite de crédito
	Efetuar crédito
	Localizar crédito

Tabela 5.1: Portfólio categorizado de serviços	
Categoria	**Serviço**
	Estornar crédito
Segurança	Registrar atividade
	Checar log de acessos
Suporte	Checar disponibilidade de recursos

5.1.4.2. Decisões arquiteturais

O portfólio de serviços nesta camada foi identificado para os processos do cenário proposto após a aplicação das técnicas de modelagem de negócio e identificação de serviços da metodologia SOMA, como decomposição de domínio e técnicas como *top-down* e *goal-to-service* descritas e detalhadas no capítulo 4, "DESENVOLVIMENTO DA ARQUITETURA".

5.1.5 Camada de Componentes de Serviços

A camada de componentes de serviços na arquitetura de micropagamentos possui uma característica diferenciada em relação às implementações SOA "tradicionais" – a presença de um componente "autônomo" – o *Autonomic Manager* – que automatiza as funções de TI dos processos que contemplam o cenário.

5.1.5.1. Áreas funcionais suportadas x componentes

Os dois componentes desta camada relacionam-se respectivamente a:

- O componente de crédito provê serviços para a área de gestão de clientes.

- O *Autonomic manager* provê automatização para as funções relacionadas à área de gestão de TI.

5.1.5.2. Domínios de negócio, objetivos e processos suportados

A área de gestão de clientes tem como objetivo prover os serviços necessários para os processos relacionados com compradores de conteúdo, internamente e externamente a organização, através de interfaces providas pelos canais providos pela camada de consumo para acesso dos próprios clientes e de funcionários internos. O processo de créditos é suportado para o cenário proposto.

O objetivo da área de gestão de TI é automatizar tarefas e procedimentos repetidos e manuais de forma a garantir os objetivos de negócio. Os processos de segurança e suporte inseridos na solução utilizam o *Autonomic Manager* para prover automação de seus serviços.

5.1.5.3. Decisões relativas a governança

Decisões relativas a governança não fazem parte do escopo do presente trabalho.

5.1.6 Camada Operacional

A camada operacional não foi utilizada na solução devido à inexistência de ativos operacionais legados.

5.1.7 Demais camadas SOA

A extensão da solução para as camadas de "Integração (ESB)", "Qualidade de Serviço", "Arquitetura de dados e Business Intelligence" e "Governança" não faz parte do escopo do presente trabalho.

6 CONCLUSÃO

Implementar uma solução ou arquitetura corporativa baseada em SOA/AC não é uma tarefa ou projeto trivial. É necessário planejamento, comprometimento estratégico da organização, envolvimento de pessoas-chave e governança da área de TI. Tecnicamente, é uma abordagem bastante diferente das tradicionais. Foca-se o negócio. Rastreia-se o negócio. Decompõe-se o negócio. Alinha-se negócios e TI. Nesse sentido, a metodologia e técnicas SOMA (*Service-oriented Modeling and Architecture*) fornecem uma excelente orientação. De qualquer forma, é necessário treinamento e amadurecimento corporativo para essa mudança.

Apesar disso, existe muita flexibilidade em sua adoção, podendo-se implementá-la de forma incremental. Outro ponto importante é a possibilidade de utilizar os ativos legados, expondo-os como serviços.

A solução apresentada para o problema de micropagamentos utilizando-se o *template SOA Solution Stack* e conceitos de *Autonomic Computing* mostra-se viável para o contexto apresentado e deve ser expandida para os demais processos de negócio. Visualizam-se os seguintes pontos de agregação:

- possibilidade de reutilização de ativos individualmente ou por composição;
- rápida implementação de novos canais ou modificação dos existentes;
- rápida implementação de novos serviços ou modificação dos existentes;
- expansão das funcionalidades de automação providas pela AC para novos processos de TI;
- redução de TCO e aumento do ROI pela introdução das características AC na solução.

Apesar do foco no problema de micropagamentos, a solução SOA/AC apresentada pode possivelmente ter bons resultados em outros tipos de negócios e plataformas, dependendo de avaliação mais aprofundada, caso a caso.

7 REFERÊNCIAS

Arsanjani, Ali. **Service-Oriented modeling and architecture**: How to identify, specify, and realize services for your SOA. IBM, 2004.

Arsanjani, Ali. et al. **Patterns: Service-Oriented Architecture and Web Services.** IBM RedBooks, 2004.

Arsanjani, Ali. et al. **Design an SOA solution using a reference architecture**: Improve your development process using the SOA solution stack. IBM, 2007.

Jr., Rob High. et al. **IBM's SOA Foundation:** An Architectural Introduction and Overview. IBM, 2005.

Ganci, John. **Patterns: SOA Foundation Service Creation Scenario**. IBM RedBooks, 2006.

Mittal, Kunal. **Requirements process for SOA projects, Part 2: Business requirements for your first SOA services**. IBM, 2006.

Portier, Bertrand. **SOA terminology overview, Part 1**: Service, architecture, governance, and business terms. IBM, 2007.

Portier, Bertrand. **SOA terminology overview, Part 2**: Development processes, models, and assets. IBM, 2007.

Portier, Bertrand. **SOA terminology overview, Part 3**: Analysis and design: Development processes, models, and assets. IBM, 2007.

Bass, Len; Clements, Paul; Kazman, Rick. **Software Architecture in Practice.** Second Edition. Pennsylvania: Addison Wesley, 2003. 528 p.

Kroll, Per; Kruchten, Philippe. **The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP.** Addison Wesley, 2003. 416 p.

Booch, Grady; Rumbaugh, James; Jacobson, Ivar. **UML: Guia do Usuário.** Editora Campus, 2000. 472 p.

Horn, Paul. **Autonomic Computing: IBM's Perspective on the State of Information Technology.** IBM, 2001.

Kephart, Jeffrey O.; Chess, David M. **The Vision of Autonomic Computing**. IEEE, 2003.

Whitepaper: An architectural blueprint for autonomic computing. IBM, 2004.

Whitepaper: Autonomic Computing: Enabling Self-Managing Solutions. IBM, 2005.

Draper, Christine. **Combine autonomic computing and SOA to improve IT management**: Learn how to incrementally delegate human-powered activities to automation. IBM, 2006.

Zimmermann, Olaf. et al. **Elements of Service-Oriented Analysis and Design**: An interdisciplinary modeling approach for SOA projects. IBM, 2004.

Johnston, Simon. **Modeling service-oriented solutions**. IBM, 2005.

Geer, David. **E-Micropayments Sweat the Small Stuff.** Computer, 2004.

Rivest, Ronald L. **Peppercoin Micropayments**. MIT, 2004.

Odlyzko, Andrew. **The Case Against Micropayments**. University of Minnesota.