Uma proposta de integração de sistemas para um ambiente de migração do ERP/SIG da plataforma COBOL para uma solução em ambiente *Web*

Luiz Fernando Maciel França, Leandro Libério Silva

Uni-BH – Centro Universitário de Belo Horizonte
Av. Prof.Mário Werneck, 1685 – Estoril – CEP: 30455-610 – BH/MG.

lfmfsig@gmail.com, leandroliberio@gmail.com

Abstract: *This article evaluates a proposal for integration between an ERP system developed using the COBOL language in desktop environment with a system developed for the WEB environment. The two systems be coexisingt in the same computing environment for a period necessary to complete the migration of desktop application for the application in the WEB environment. The proposed integration is based on the construction of middlewares. The solution presented and tested through prototype shown to be effective given the expectations of the team to deal with migration.*

Resumo: *Este artigo avalia uma proposta de integração entre um sistema ERP desenvolvido utilizando a linguagem COBOL em ambiente desktop com um sistema desenvolvido para o ambiente WEB. Os dois sistemas coexistirão em um mesmo ambiente computacional por um período necessário para a migração completa da aplicação desktop para a aplicação no ambiente WEB. A proposta de integração é baseada na construção de middlewares. A solução apresentada e testada por meio de protótipo demonstrou ser eficiente atendendo as expectativas da equipe de responsável pelo processo de migração.*

1.Introdução

O ERP/SIG é um sistema de gestão desenvolvido pela empresa Sig Informática Ltda., que foi criado para atender as necessidades de micro e pequenas empresas. O início de seu desenvolvimento foi em 1985. Naquele período não havia o conceito de sistemas de gestão integrada e a proposta era somente automatizar processos específicos das organizações, como emissão de notas fiscais, faturas/duplicatas e um controle financeiro bem simplório.

Em 1995, a partir de uma parceria estabelecida com uma empresa de consultoria organizacional e empresarial, se passou a focar a automatização não mais como fim, mas como meio para auxiliar os processos decisórios e o sistema passou a ter um enfoque gerencial.

O ERP/SIG foi desenvolvido utilizando a linguagem COBOL abordada no capítulo 2 item 2.2, criada a partir da iniciativa do Pentágono (USA), para estabelecer uma linguagem comum de programação para computadores digitais[Bastos 1983].

O objetivo deste trabalho é a avaliação de uma proposta de integração de uma base de dados no formato ISAM (*Indexed Sequential Access Method* ou Método de

Acesso Seqüencial e Indexado) com uma base de dados relacional, através da construção de dois *middlewares* (programa de interface), com isto espera-se que as bases mantenham-se sincronizadas e consistentes, proporcionando uma migração de forma gradativa, sem criar grandes transtornos aos clientes que possuem o ERP/SIG instalado. Um dos dois *middlewares,* foi desenvolvido utilizando a linguagem COBOL e recebe através de XML (*eXtensible Markup Language* ou Linguagem de marcação estendida) os dados que atualizados no novo sistema deverão ser replicados para base ISAM. O segundo *middleware* foi desenvolvido utilizando a linguagem *Python* e tem o objetivo de receber via XML as modificações que ocorrerem na base ISAM e que deverá ser replicado para nova base de dados relacional.

2 Fundamentação teórica

Neste capítulo, apresentaremos os principais conceitos e tecnologias para a realização da migração/evolução do ERP/SIG para uma solução em ambiente *Web*.

Na seção 2.1, veremos sobre a evolução das aplicações legadas. Na seção 2.2, apresentaremos a plataforma COBOL utilizada, detalhando os recursos e paradigmas da ferramenta. Na seção 2.3, teremos uma abordagem sobre ERP. Na seção 2.4, apresentaremos o *Python* com seus recursos e paradigmas. Na seção 2.5, apresentaremos o XML. Na seção 2.6, faremos uma abordagem sobre *Middleware*.

2.1 A evolução das aplicações legadas

A obsolescência das aplicações acontece por elas terem sido criadas para atender a necessidades específicas de uma época, que, muitas vezes, não representa mais as necessidades atuais. Além disto, as tecnologias utilizadas podem se encontrar defasadas, e muitas vezes já não são suportadas por seus fabricantes [Fontanette 2005].

O dinamismo existente no mundo dos negócios não permite que as empresas abram mão de seus sistemas computacionais, sob pena de não conseguirem se sustentar no mercado. Desta forma, o processo evolutivo destas aplicações se faz necessário utilizando novos conceitos de negócios, de *software* e a utilização de novas tecnologias no seu desenvolvimento.

Para substituir um *software* ou desenvolvê-lo novamente, é necessário a reconstrução da aplicação desde o início. Isto é apropriado quando o *software* não consegue atender às necessidades do negócio e quando a modernização não é possível ou não vale a pena em relação aos custos. A este tipo de sistema dá-se o nome de "aplicações legadas" . [Fontanette 2005].

2.2 COBOL

A linguagem COBOL foi criada em 1959 numa iniciativa do Pentágono (USA) que objetivava estabelecer uma linguagem comum de programação para ser utilizada na solução de problemas comerciais [Bastos 1983].

COBOL é uma linguagem de 3ª geração, e é padronizada pelo *American National Standards Institute* (ANSI)[1] desde 1968. É composta por quatro divisões onde

[1] O *American National Standards Intitute* é a instituição responsável pela analise das necessidades de novas implementações para a linguagem COBOL. Periodicamente se reúnem com grupos de usuários

cada uma tem um papel específico dentro da linguagem, conforme figura 1. Estas divisões auxiliam na simplicidade, eficiência e leitura do programa. Os objetivos destas divisões são:

IDENTIFICATION DIVISION. Divisão de identificação, nela são identificados, o autor, programa-fonte e módulo-objeto.

ENVIRONMENT DIVISION. Especifica o equipamento a ser utilizado bem como os periféricos envolvidos no funcionamento do programa.

DATA DIVISION. Descreve os dados que o programa aceitará como entrada e os que serão produzidos como saída, bem como todos os dados intermediários utilizados no programa.

PROCEDURE DIVISION. Descreve todas as ações a serem tomadas pelo programa, é nesta divisão que se desenvolve a lógica do programa.

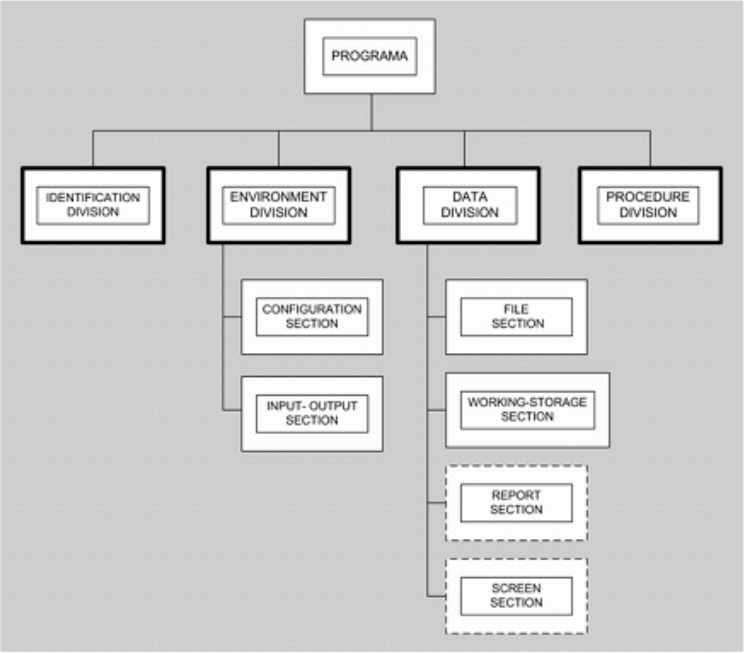

Figura 1 – Estruturas das divisões de um programa COBOL [Fontanette 2005]

O ERP/SIG foi desenvolvido utilizando o RM/COBOL, compilador disponibilizado pela *Liant Software Corporation*. O RM/COBOL gera representações intermediarias dos aplicativos em COBOL. Para executar as aplicações compiladas pelo RM/COBOL, é necessário um interpretador. Esse interpretador é comumente conhecido como *runtime*[2], permitindo com isto que um mesmo programa possa funcionar em diversas plataformas, bastando para isto instalar um *runtime* adequado a cada plataforma.

---

e fabricantes, e determinam novos padrões para a linguagem mantendo-a sempre atualizada junto às necessidades do mercado.

[2] *Runtime* – Conjunto de programas ou bibliotecas necessário à execução de programas gerados para a linguagem específica. O *runtime* é semelhante a uma máquina virtual.

A maioria dos compiladores até algum tempo atrás, implementava como forma de persistência de dados, estrutura de base de dados ISAM[3]. Raros eram os compiladores que trabalhavam com bancos de dados relacionais. A implementação da base de dados ISAM normalmente é feita em dois arquivos, um arquivo de dados e outro com a estrutura de índices. O RM/COBOL implementa a base ISAM de uma forma diferente, utiliza um mesmo arquivo para a base de dados e para a área de índices.

Atualmente, a linguagem COBOL, tem recebido diversas implementações como, tecnologia para acesso a bases de dados relacionais como *MySQL*, *Firebird*, *PostGreSQL*, além de tecnologias para desenvolvimento *Web* [Price 2001]. Apesar da padronização oferecida pelo ANSI (*American National Standards Institute*), cada fornecedor, implementa recursos que considera importante do ponto de vista comercial, recursos ainda não padronizados pelo ANSI, o que gera incompatibilidades entre os diversos dialetos COBOL.

2.3 ERP

Na década de 90, com a evolução dos computadores e dos *softwares*, tornou-se possível a integração de diversos módulos de programas existentes nas empresas que atendiam as demandas de manufatura, finanças, logísticas, gestão de RH[4] e vendas em conjunto com marketing. Com isto surgiram os sistemas atuais de ERP (*Enterprise Resource Planning* ou Planejamento de Recursos Empresariais), que oferecem às organizações, não apenas uma solução que contempla informações para apoio a decisão gerencial no âmbito dos sistemas produtivos, mas que também contempla a integração interna da empresa e externa com seus fornecedores, clientes e outros parceiros de negócio [Freitas Silva 2004]

Os sistemas ERPs são sistemas de gestão empresarial caracterizados principalmente por abranger um amplo escopo de funcionalidades, pela integração de seus dados e pela capacidade de adaptação a vários tipos de organização [Zancul 2000].

As funcionalidades dos sistemas ERP são geralmente organizadas em módulos referentes às áreas funcionais das empresas. O agrupamento das funcionalidades em módulos e a denominação dada a cada um dos módulos é arbitrária e definida pelos fornecedores desses sistemas. De uma forma geral, os módulos dos sistemas ERP abrangem o seguinte escopo: Finanças e contabilidade, vendas e gerenciamento do relacionamento com clientes, planejamento da produção, compras e gestão de suprimentos, gerenciamento de projetos, recursos humanos, gestão da qualidade, entre outros [Zancul 2000].

Os sistemas integrados melhoram o fluxo de dados nas empresas e facilitam o acesso a informações gerenciais, resultando, na maioria das vezes, em enormes ganhos de produtividade e em maior velocidade de resposta [Davenport 1998].

---

[3] ISAM – *Indexed Sequential Access Method* – Método para armazenamento de dados que proporciona a recuperação da informação de forma bem rápida.

[4] RH – Recursos Humanos – Um sistema de gestão de recursos humanos é composto por módulos de gestão de pessoas, candidatos, folha de pagamento, controle de ponto, etc.

2.4 *Python*

A linguagem *Python* foi criada por Guido Van Rossum em 1991 e seu desenvolvimento foi em grande parte derivado do ABC, uma linguagem didática que foi desenvolvida nos anos 80. Apesar disto, *Python* foi concebido para resolver problemas reais e tomou emprestada uma grande quantidade de características de linguagens de programação como C++, Java, Modula-3 e Scheme. [Downey 2002].

A linguagem é utilizada em diversas áreas como servidores de aplicação, computação gráfica e ambiente *Web*. É naturalmente orientada a objetos ainda que suporte outros paradigmas, como a programação funcional e modular, é interpretada e multi-plataforma, roda em Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm *Handhelds*, e telefones celulares Nokia.

Para a linguagem, foram desenvolvidos vários *frameworks*[5] para desenvolvimento tais como o Plone, ZOPE, Django e Turbogears.

O *Python* possui uma licença compatível com a GPL *(General Public License* ou Licença Pública Geral) e seus direitos são de propriedade da *Python Software Foundation*.

Algumas características do *Python*[Python 2007]:

- grande variedade de tipos básicos de dados, números (ponto flutuante, complexos, inteiros de tamanho ilimitado), *strings* (ASCII e Unicode), listas e dicionários;
- programação orientada a objetos, com herança múltipla de classes;
- agrupamento de código em módulos ou pacotes;
- suporta tratamento de exceções *raising and catching;*
- tipo de dados fortemente tipados onde os tipos incompatíveis produzem exceções;
- *Python* possui procedimentos de programação avançados como "*generators*" e "*list comprehensions*" e
- *Python* libera o uso de memória automaticamente sem que você tenha que controlar manualmente em seu código.

2.5 XML

Em 1969 Ed Mosher, Ray Lorie e Charles F. Goldfarb, da IBM *Research*, inventaram a primeira linguagem de marcação geral (GML), era uma linguagem que tinha a proposta de ser uma meta-linguagem[6]. A GML se tornou mais tarde SGML (*Standard General Markup Language* ou Linguagem de marcação geral padrão) [Anderson 2001].

Em 1996, a XML (*eXtensible Markup Language* ou linguagem de marcação estendida) teve seu projeto iniciado pelo *World Wide Web Consortium* (W3C) que tinha o objetivo de criar uma linguagem com a flexibilidade e capacidade do SGML e com a ampla aceitação do HTML (*Hipertext Markup Language* ou Linguagem de Marcação de Hipertexto) [Anderson 2001].

---

[5] Ferramentas projetadas com a intenção de facilitar o desenvolvimento de *software.*

[6]Uma linguagem usada para descrever outras linguagens, suas gramáticas e seus vocabulários

A XML em si é muito simples, são as linguagens filhas que a torna poderosa [Castro 2001]. Basicamente, é um arquivo ASCII, que contém *tags* auto-descritivas criadas pelo próprio desenvolvedor. Estas *tags* (marcas) tem o objetivo de identificar o que é a informação, e não a sua apresentação, como acontece com o HTML. Por este motivo, é muito utilizada para transferir dados entre fontes de tecnologias diferentes. Apesar da flexibilidade de criação de suas próprias *tags*, existem regras bem rígidas quanto à estrutura do documento XML.

2.6 *Middleware*

Middleware é um termo utilizado para designar camadas de *software* que não constituem diretamente aplicações, mas que facilitam o uso de ambientes ricos em tecnologia da informação. Ajuda a reduzir a complexidade do negócio, uma vez que se transferem para o *middleware* funções que não são o objeto principal da aplicação. A camada de *middleware* pode proporcionar serviços como comunicação inter-programas tais como autenticação, identificação entre outras.

Middleware é um componente de *software* que tem como finalidade interligar processos clientes a processos servidores, disponibilizando um conjunto de serviços que visam reduzir a complexidade do processo de desenvolvimento de uma aplicação. O *midlleware* pode ser visto como uma ferramenta que salva os desenvolvedores de se envolver em detalhes de cada fonte de dados. Por esta definição, o termo *middleware* é utilizado para descrever diversos níveis de interoperabilidade[7], desde os níveis de abstração mais baixos, como em aplicações mais próximas ao sistema operacional e do sistema de rede, até os sistemas complexos como nos casos de sistema de integração de dados com maior nível de abstração[Barbosa 2001].

3 Desenvolvimento

3.1 ERP/SIG

O ERP/SIG foi concebido para utilizar interface CUI (*Caracter User Interface* ou Interface de Usuário por Caractere) e base de dados ISAM. Inicialmene o ERP/SIG contemplava módulos que trabalhavam de forma separada e independente. A partir de 1995 seus módulos começaram a ser integrados para atender a necessidades organizacionais.

Desde a sua concepção, este sistema vem sofrendo modificações e adaptações, para atender as novas necessidades e novos direcionamentos de negócios. O ERP/SIG possui aproximadamente 2.500 programas com mais de 1.000.000 linhas de código de acordo com análise interna realizada em 27/09/2007.

Os principais módulos deste sistema são: financeiro, compras, vendas, faturamento, estoque/almoxarifado, escrituração fiscal e gerência. Estes módulos são divididos em sub-módulos. Na figura 2 temos um exemplo de tela da funcionalidade "Manutenção e clientes".

---

7 Interoperabilidade – Capacidade de compartilhar e trocar informações entre sistemas heterogêneos e independentes.

Para o novo sistema, definiu-se utilizá-lo em plataforma *Web* através de *browsers* (navegadores), com utilização de um sistema gerenciador de banco de dados baseado num projeto de base de dados com visão integral do sistema, e que mantenha todas as funcionalidades já existentes, porém construído numa nova arquitetura que permitirá atualizações de forma mais rápida e segura.

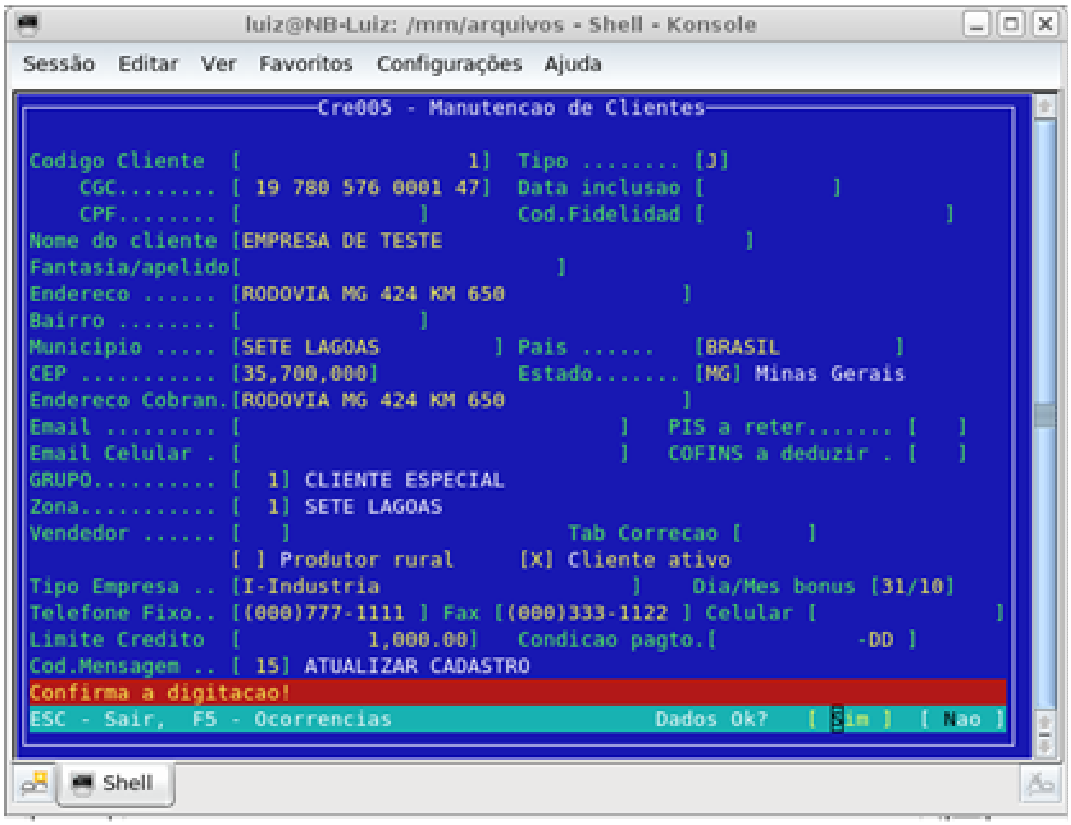

Figura 2 – Tela de Manutenção de Cadastro de Clientes SIG/ERP

3.1.1 – *Middleware* COBOL-ISAM

A partir de 1995, junto com as alterações realizadas no ERP/SIG para integração de seus módulos, também foi criado um *middleware* para cuidar da atualização da base de dados e da geração de *logs*[8] das transações. A idéia deste *middleware* partiu da necessidade de garantir que toda transação ocorrida na aplicação fosse registrada em *logs* de maneira a permitir em qualquer tempo, auditoria para identificação de possíveis problemas (operacionais ou de sistema) na aplicação. Desta maneira, todo programa que precisa atualizar qualquer informação na base de dados, executa este *middleware*, para que o mesmo faça a atualização e o registro dos *logs*.

O *middleware* COBOL-ISAM não fica instanciado permanentemente em memória. Ele somente é executado quando chamado pelos programas do ERP/SIG. Neste momento ele executa o que foi pedido, retorna o status da operação para a aplicação e em seguida se encerra. A figura 3 ilustra o posicionamento deste *middleware* na aplicação.

[8] São arquivos que contém os registros deoperações realizadas.

Além dos *logs* de transação, este *middleware* também gera *logs* de erro caso não consiga fazer as transações solicitadas.

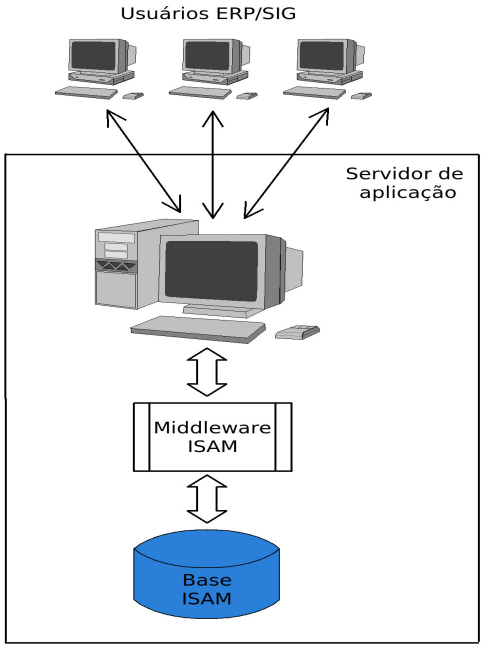

Figura 3 – Middleware COBOL-ISAM

3.2 – Proposta de migração

O novo sistema poderia ser reescrito utilizando a própria linguagem COBOL, que vem passando por atualizações tecnológicas, e atualmente oferece todos os recursos necessários ao novo desenvolvimento. Nas novas versões do COBOL, já existem implementações para desenvolvimento de aplicações para o ambiente *Web*, para utilização de banco de dados relacionais além de outras inovações. Uma das grandes dificuldades para isto é o alto custo do compilador e de suas ferramentas, bem como a necessidade de se adquirir *runtime* para ser instalado junto a cada computador onde for instalado uma cópia do ERP/SIG, onerando o produto e dificultando a sua colocação no mercado.

Então optou-se por uma proposta de migração gradativa, auxiliada pelo espelhamento de base de dados ISAM e relacional por meio de *middlewares* que serão descritos nas próximas seções. Sendo que para análise e posterior constatações utilizamos o protótipo de uma das funcionalidades.

3.2.1 – Especificação da nova plataforma adotada

Baseado no propósito de desenvolvermos um produto que contemplasse as novas necessidades e não incorporasse nele custo de produtos de terceiros, foi estudado pela equipe responsável pelo ERP/SIG diversas linguagens para desenvolvimento *Web* como *Python*, PHP, Java e Ruby. A escolha por *Python* se deve a diversos fatores já expostos no capítulo 2 item 2.4 bem como a outros fatores ligados a preferência da equipe de desenvolvimento. O SGBD a ser utilizado será o MySQL. Sua escolha se deu pela experiência que a equipe já possui com este SGBD, por utilizá-lo em outras aplicações da empresa.

Para desenvolvimento completo da nova aplicação utilizando linguagem *Python*, estima-se que serão necessários entre 2 a 3 anos de planejamento/desenvolvimento. Durante este período, à medida que módulos ou sub-módulos do sistema vão sendo concluídos, estes módulos já poderão ser instalados nos clientes, iniciando assim o processo de migração de forma gradativa.

Esta migração gradativa é proposital, pois minimizará o impacto da transição sob dois aspectos. Primeiro no que diz respeito ao cliente e ao ambiente operacional do mesmo, com treinamento dos funcionários que utilizarão o novo sistema, além de não criar a necessidade de contratação de novos funcionários gerando custos desnecessários. O segundo aspecto diz respeito à própria equipe da SIG que também não terá a necessidade de contratação de novos funcionários para as atividades de suporte e treinamento.

Para que esta migração gradativa possa acontecer, será necessário fazer a integração dos dois sistemas, de tal maneira que todas as transações acontecidas em um sistema sejam atualizadas no outro sistema de forma a gerar esta redundância proposital.

Durante todo o período de desenvolvimento, até a sua conclusão, os dois sistemas conviverão no mesmo ambiente de produção. A versão atual (COBOL), com todos os seus módulos e a nova versão (Python) com os módulos e sub-módulos já concluídos, substituindo gradativamente os módulos do antigo ERP/SIG. A figura 4 ilustra o ambiente com as duas aplicações.

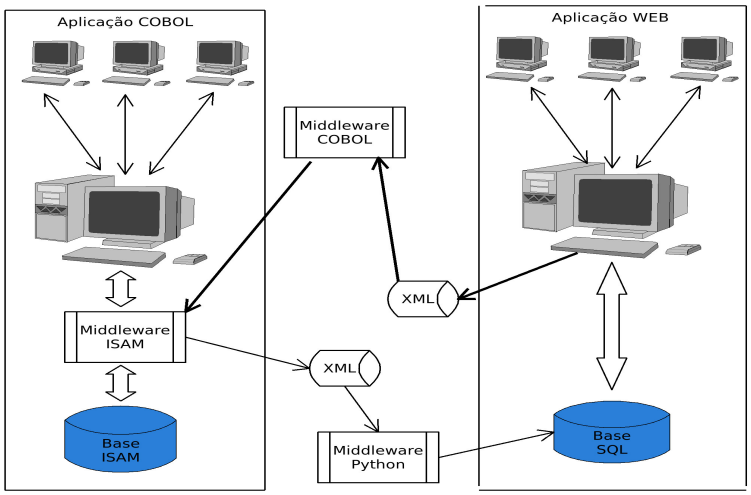

Figura 4 – ambiente operacional.

Todos os dados estarão replicados em duas bases distintas, uma ISAM alimentada pelo antigo sistema e a outra relacional alimentada pelo novo sistema, permitindo que os sistemas funcionem independentes porém integrados. Considerando ser apenas no período de desenvolvimento/migração, optou-se pela replicação da base de dados por permitir maior segurança ao negócio dos clientes.

A replicação será apenas da base de dados, e não das operações realizadas nos sistemas, uma vez que cada tarefa será feita apenas uma vez, no ERP legado ou no ERP *Python*, para os módulos que já estiverem substituindo o legado.

Com os recursos atuais do compilador COBOL utilizado, não é possível acessar a base relacional gerada pela nova aplicação a ser desenvolvida em *Python*, assim como a nova aplicação não terá recursos para acessar a base de dados ISAM gerada pelo RM/COBOL. Assim, para que as duas aplicações possam conviver em um mesmo ambiente pelo tempo necessário para realização de toda migração há que se criar uma solução intermediária para resolver este problema. A solução proposta é a criação de dois *middlewares*, um utilizando a linguagem COBOL, que será chamado de *middleware* COBOL-XML que fará a comunicação com o *middleware* COBOL-ISAM e pela criação de um *middleware* em *Python* que será chamado de *middleware Python-XML* que fará a atualização de dados na base relacional. Estes dois *middlewares*, COBOL-XML e *Python*-XML, diferentemente do *middleware* COBOL-ISAM serão aplicações que ficarão residentes em memória, similar a um serviço[9] e que serão responsáveis pela atualização de suas respectivas bases de dados através de troca de arquivos XML residente via sistema de arquivos em repositórios pré-definidos. Ver figura 5.

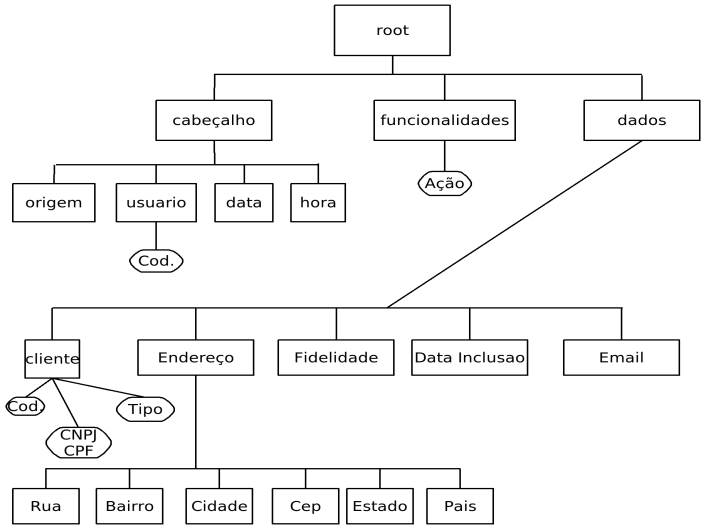

Figura 5 – Diagrama XML

Cada transação realizada no ERP/SIG gera atualização em uma ou mais tabelas/arquivos e para cada transação será gerado um arquivo XML, de tal maneira que em um arquivo XML poderá haver dados referentes a instruções para atualização de uma ou mais tabelas/arquivos. Em função da quantidade de usuários utilizando o sistema de forma simultânea, poderão existir vários arquivos XML aguardando para serem utilizados pelo *middleware* responsável. À medida que os arquivos XML são utilizados pelos *middlewares*, deverão ser removidos do diretório, pois não serão mais necessários.

3.2.2 – *Middleware* COBOL-XML

Pelo fato da implementação da base de dados ISAM utilizada ser proprietária, aplicações desenvolvidas utilizando outras linguagens que não sejam COBOL e que não

[9] Serviços – Programas que ficam funcionando sem a intervenção do usuário para realizar atividades específicas.

sejam o COBOL da Liant (RM/COBOL), não conseguem atualizar bases de dados ISAM. Assim, o novo ERP/SIG desenvolvido em *Python* não conseguirá atualizar na base ISAM do antigo ERP/SIG como se faz necessário. Para atender a esta necessidade, o novo ERP/SIG deverá gerar um arquivo XML com os dados a serem alterados e entregará este arquivo para o *middleware* COBOL-XML. O *middleware* COBOL-XML, ficará residente em memória e terá como função analisar um diretório pré-estabelecido, verificando a existência de arquivos XML para que possa lê-los e em seguida chamar o *middleware* COBOL-ISAM que fará atualização na base de dados ISAM bem como fará a geração dos *logs* necessários. Concluída sua tarefa o *Middleware* COBOL-ISAM, será encerrado e retornará o status da operação para o *middleware* COBOL-XML que voltará a verificar o diretório base a procura de novos arquivos XML para atualização de outras transações.

Após a conclusão de cada transação, o *middleware* COBOL-ISAM deverá gerar *log* da transação realizada, bem como *log* de erro caso não consiga realizar a transação juntamente com um histórico do fato ocorrido.

3.2.3 – *Middleware Python-XML*

O atual ERP/SIG escrito em COBOL, por limitação da versão do COBOL, não tem recursos para acessar a base de dados relacional. A única forma disponível nas condições atuais é gerar um arquivo texto do tipo XML para que uma aplicação que consiga acessar a base relacional possa fazê-lo. Esta será a função do *middleware Python-XML*.

O *middleware Python-XML* ficará residente em memória e terá como função, analisar um diretório pré-estabelecido verificando a existência de arquivos XML para que possa lê-los e em seguida atualizar a base relacional. Concluída sua tarefa de atualização da base relacional, ele voltara a pesquisar o diretório base a procura de outros arquivos XML referente a novas transações.

Da mesma forma que o *middleware* COBOL-XML, o *middleware Python-XML* deve, após a conclusão de cada transação, gerar *log* da transação realizada, bem como *log* de erro caso não consiga realizar a transação juntamente com um histórico do fato ocorrido.

3.2.4 – Roteiro de migração para a nova plataforma

Para efetuar a migração criamos um programa COBOL que lerá a base ISAM e gerará um arquivo texto contendo o SQL para popular inicialmente a base de dados relacional.

Após a geração do arquivo SQL, será feita a sua importação no SGBD usando as próprias bibliotecas do SGBD.

3.3 – Protótipo

Construímos um protótipo para confirmar-mos a proposta apresentada neste artigo. O protótipo foi criado tomando por base uma única função existente no ERP/SIG, que foi a função de “manutenção de clientes”. Para alcançar este objetivo as seguintes ações foram necessárias:

1. criação de programa COBOL para ler a base ISAM e gerar um arquivo texto contendo o SQL. Esta ação faz uma copia da base de dados ISAM para a base

relacional, e é necessária para iniciar o processo de integração. Esta ação é realizada apenas uma vez com o objetivo de popular a base relacional;

2. criação utilizando a linguagem *Python* da funcionalidade "manutenção de clientes" para que seja possível alimentar a base relacional. Na Figura 6 vemos a tela da funcionalidade implementada em *Python*;
3. criação de dois *middlewares* para manter as bases integradas, um em COBOL e o outro em *Python*;
4. criação massa de testes com inclusão, exclusão e alteração para a base ISAM e outra para a base relacional com 100 transações para cada e
5. verificação nos dois sistemas se a massa de testes foi implementada conforme esperado.

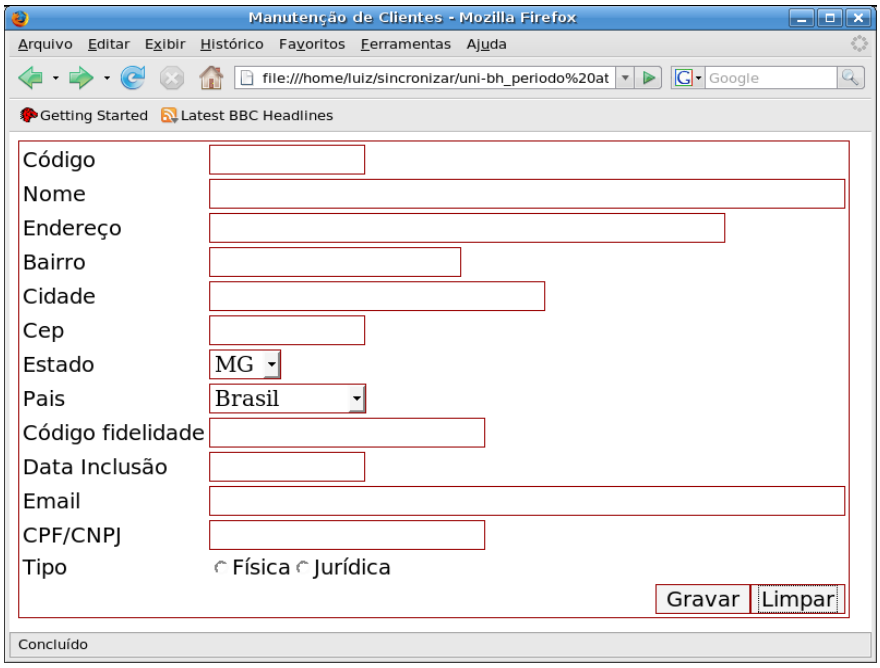

Figura 6 – Tela de Manutenção de Clientes desenvolvido em *Python* – Ação 3

Executamos as etapas 4 e 5 por vinte vezes e apuramos a media do tempo para atualização das bases de dados que foi de menos de 0,1 segundo para 100 funcionalidades.

4 – Conclusão

O objetivo deste trabalho foi o estudo de uma metodologia para integração de sistemas em um ambiente de migração, onde o primeiro desenvolvido em COBOL com base de dados ISAM será gradativamente substituído pelo segundo desenvolvido em *Python* para ambiente *Web* utilizando um SGBD como base de dados. Enquanto a migração não se concluir os dois sistemas terão que conviver de forma harmoniosa, sem gerar problemas e retrabalho para seus usuários. A proposta de construção de dois *middlewares* COBOL-XML e *Python*-XML para manter as duas bases de dados ISAM e relacional sincronizadas se mostrou muito eficiente.

Esta alternativa apresentou as seguintes vantagens:

- pouca modificação no antigo ERP/SIG, permitindo à equipe de desenvolvimento se concentrar no desenvolvimento do novo sistema;
- integração total da aplicação permitindo inclusive que a mesma funcionalidade possa ser usada simultaneamente nos dois sistemas;
- segurança no processo de migração, permitindo aos usuários voltarem atrás no processo de migração, caso ocorra algum problema com o novo sistema e
- rápida atualização da base de dados nos dois sistemas em função da solução ser independente das aplicações.

Alguns pontos negativos também foram observados, apesar de já serem esperados pela equipe. São eles:

- necessidade do cliente ter que usar duas aplicações distintas em ambientes distintos até a conclusão da migração;
- possíveis conflitos de dados no momento da transferência de arquivos XML e
- redundância de dados causada pelo espelhamento.

5 – Trabalhos futuros

O *middleware* construído para este trabalho possui um alto acoplamento com a aplicação, não tendo sido criado para atender a outras integrações que não esta contemplada neste trabalho. Para tanto sugerimos o estudo de tecnologias para criação de *middlewares* genéricos como:

- Estudo da estrutura da base de dados ISAM;
- estudo de ferramentas ORM (*Object Relational Mapper*) e
- definição de padrões de arquivos XML para informação aos *middlewares* da estrutura da base de dados ISAM.

Com o estudo destas tecnologias e definição de padrões, acreditamos ser possível a criação de *middlewares* genéricos que possam atender a qualquer integração de aplicações que utilizam base de dados ISAM com aplicações que utilizam SGBDs.

Referência bibliográfica.

Anderson, Richard et al. **Professional XML.** Rio de Janeiro: Ciência Moderna, 2001. 1266 p. ISBN 8573931167

Barbosa, Álvaro César Pereira (2001). ***Middleware* para integração de dados heterogêneos baseado em composição de frameworks.** Pontifícia Universidade Catolida do Rio de Janeiro.

Bastos, Alex C. (1983), **Programação COBOL**, 4ª edição

Castro, Elizabeth. **XML para a *World Wide Web*.** Rio de Janeiro: Campus, ©2001. 269 p. (Visual quickstart guide) ISBN 8535207384.

Davenport, Thomas H. 1998. ***Putting the Enterprise into the Enterprise System*** . Harvard Business Review.

Downey,Allen B., Elkner, Jeffrey e Meyers, Chris. (2002) ***How to Think Like a Computer Scientist.Learning with Python***. Disponível em http://www.ibiblio.org/obp/thinkCSpy em 28/09/2007 - Segunda edição.

Fontanette, Valdirene (2005), **AMGra: Uma abordagem para migração gradativa de aplicações legadas** – Universidade Federal de São Carlos.

Freitas Silva, Silvio (2004). **Proposta de Modelo de Sistemas de Gestão Integrada ERP para Pequenas e Medias Empresas.** Faculdade de Engenharia Mecânica . Universidade Estadual de Campinas.

Python (2007), Informações capturadas da página: http://www.python.org, Setembro.

Price, Wilson. (2001) ***Elements of COBOL Web Programming with Micro Focus Net Express*.** Object-Z Publishing, ISBN 0965594513.

Zancul, Eduardo de Senzi (2000), **Análise da aplicabilidade de um sistema ERP no processo de desenvolvimento de produtos** – Escola de Engenharia de São Carlos – Universidade de São Paulo.