AMGraA: Uma Abordagem para Migração Gradativa de Aplicações◊

Valdirene Fontanette[1], Antonio Francisco do Prado[1], Marco Antonio Pereira[1], André Luis Costa de Oliveira[2]

[1]Departamento de Computação - Universidade Federal de São Carlos (UFSCar)
Rod. Washington Luís, Km 235 - Caixa Postal 676 - Cep.13565-905 - São Carlos-SP

[2]Apyon Technology S/AAv. Maria Coelho Aguiar, 215 Bloco G – Piso Jardim - Cep. 05805-000 - São Paulo - SP

{valdirene,prado}@dc.ufscar.br, pemarco@gmail.com, andré.oliveira@apyon.com.br

Resumo. *Novos requisitos, mudanças de tecnologias, legislações e outras necessidades exigem que as aplicações sejam atualizadas ao longo do seu tempo de vida. Pesquisas têm sido realizadas para oferecer suporte a essa evolução contínua do software. Hoje, essa evolução é ainda maior considerando as novas tecnologias para Web e para a computação ubíqua. Embora existam várias abordagens de modernização de aplicações na literatura, o processo nas empresas ainda é realizado quase sempre de forma manual e "ad-hoc". Motivados em pesquisar e melhorar esse processo de reconstrução de software apresenta-se uma abordagem, denominada AMGraA, para a modernização gradativa do software.*

Abstract. *New requirements, changes of technologies, legislations and other needs demand that the applications are updated along their lifetime. Researches have been accomplished to offer support to this continuous evolution of the software. Today, that evolution is even higher considering the new Web technologies and the ubiquitous computation. Although there are several approaches that support the modernization process in the literature, this is almost always accomplished manually and ad-hoc by the companies. Motivated to research and improve that software reconstruction process it is presented an approach, called AMGraA, for a gradual software modernization.*

◊ Projeto Financiado pelo Programa de Apoio à Inovação Tecnológica em Pequenas Empresas (FAPESP - PIPE)

1. Introdução

Uma aplicação de *software* é um artefato evolutivo e, com o decorrer do tempo, seu projeto e implementação originais são modificados para atender a novos requisitos e/ou melhorar o seu desempenho, incorporando conhecimentos substanciais do seu contexto. A esse processo dá-se o nome de "Manutenção de *Software*". Mas, a partir de certo período de tempo, as aplicações tornam-se obsoletas, pois a tecnologia em que as mesmas foram desenvolvidas torna-se antiga e, na maioria das vezes, não têm mais o suporte de seus fabricantes, necessitando, portanto, ser modernizada. A essas aplicações, dá-se o nome de aplicações legadas. Nestes casos, tem – se a "Modernização de *Software*" [Seacord 2003] que envolve mudanças mais extensas que a manutenção, e conserva uma porção da aplicação existente. Ela envolve a reestruturação da aplicação e a inserção de novas funcionalidades, no qual novas tecnologias e novas práticas são aplicadas.

A evolução dos negócios de uma empresa ao longo dos anos requer uma evolução sincronizada de suas aplicações legadas, porém, tais aplicações deveriam sempre oferecer um nível de qualidade adequado, de forma que estas pudessem ser mantidas facilmente. Infelizmente, devido à degradação, as aplicações legadas muito freqüentemente possuem baixos níveis de qualidade, pouca documentação ou documentação desatualizada, e, como conseqüência, suas manutenções ficam muito caras [Bianchi and Visaggio 2003]. Essas aplicações são normalmente essenciais para o bom funcionamento das empresas ou até mesmo críticas e reconstruí-las usando novas tecnologias é uma boa solução, ao invés de construir uma nova aplicação do início, o que geralmente envolve riscos e um processo complexo e trabalhoso.

Há várias abordagens e técnicas na literatura, que apóiam o processo de modernização de aplicações, visando decompor aplicações, manipular, analisar [Systa 1999], sintetizar [Biggerstaff 1994], componentizar [Álvaro 2003] e visualizar artefatos de *software* [Price 1993]. Entretanto, estas abordagens e ferramentas, ainda são complexas para o uso amplo na indústria, pois exigem alto conhecimento técnico e possuem algumas limitações, como a falta de flexibilidade no processo de migração e de integração com outras ferramentas, fazendo com que a empresa ao adotá-las perca investimentos realizados anteriormente. Portanto, embora existam, na literatura, várias abordagens para modernização de aplicações, essa tarefa nas empresas ainda é realizada quase sempre de forma manual e "ad-hoc".

A migração de uma aplicação legada pode ser feita de várias formas, porém, todas exigem diferentes mecanismos e controles. Um dos requisitos é que a aplicação legada precisa ser migrada aos poucos e ao mesmo tempo precisa continuar funcionando. Desta forma, em muitos casos é necessário adicionar novas "camadas" na aplicação para que haja integração entre a nova camada construída e a aplicação legada. Assim, a substituição é feita aos poucos, o resultado é mais rápido, e o usuário da aplicação não sente tanto os erros envolvidos em uma migração, e que causam problemas, acarretando um longo tempo de espera, devido à má compreensão do legado e a falta de funcionalidades.

Motivados por esses problemas, pela necessidade das empresas em migrar suas aplicações de forma gradativa e com o suporte de ferramentas que auxiliam o processo de migração, o GOES (Grupo de Engenharia de *Software*), juntamente com a Apyon Technology, empresa atuante no desenvolvimento de software, apoiados pela FAPESP

[Oliveira 2004], [Fontanette 2004], pesquisaram e definiram uma abordagem para migração gradativa de aplicações legadas. A abordagem proposta, através da identificação e separação dos modelos de negócios da tecnologia utilizada na aplicação legada, fornece para as empresas diretivas para o planejamento de diversas estratégias de modernização, respeitando os interesses, necessidades e investimentos de cada empresa.

Este artigo esta organizado da seguinte maneira: a seção 2 apresenta a abordagem AMGraA, a seção 3 apresenta um estudo de caso realizado com a aplicação legada Ômega, a seção 4 relata alguns trabalhos correlatos, a seção 5 apresenta uma análise crítica e limitações encontradas e a seção 6 apresenta as conclusões alcançadas.

2. AMGraA: Uma Abordagem para Migração Gradativa de Aplicações

Considerando as experiências no projeto de Reengenharia de *Software* usando Transformações (RST) [Prado et al. 2001], [Fontanette 2002] e baseado nos estudos realizados, foi definida a AMGraA: uma abordagem para a migração gradativa de aplicações legadas. A AMGraA baseia-se na separação do negócio da aplicação da sua parte tecnológica. Esta separação permite uma migração gradativa, reduzindo a complexidade envolvida nos processos de modernização e também reduzindo o tempo da migração, uma vez que os resultados são acompanhados gradativamente, conforme a migração da aplicação.

A abordagem compreende 6 fases: Avaliar Aplicação Legada; Construir Domínio e Transformadores; Analisar Aplicação Legada; Planejar Projeto de Migração; Migrar Aplicação Legada e Validar Migração da Aplicação. As duas primeiras fases são fases preparatórias para apoiar a realização da migração.

O Engenheiro de Software é apoiado por três ferramentas, que, integradas, auxiliam na execução da abordagem e compõem um ambiente para a modernização de aplicações legadas, conforme mostra a Figura 1. São elas: o sistema de transformação ST Draco-PUC [Prado et al. 2001], o Apyon Studio [Oliveira 2004], ambiente de desenvolvimento de aplicações em alto nível de abstração e a ferramenta Migration Project Planning (MPP), construída, neste trabalho, para complementar o processo, auxiliando na apresentação dos dados do modelo de negócio e no planejamento da migração.

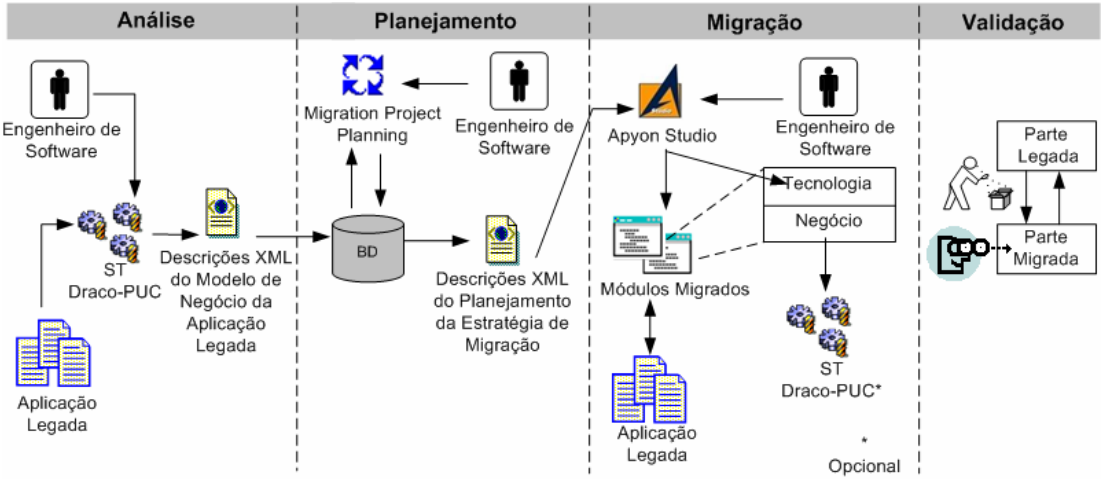


Figura 1 – Modelo de Execução da Abordagem

Ainda na Figura 1 tem-se o modelo de execução da realização da migração. A integração das ferramentas é realizada através de arquivos XML em duas fases da abordagem, na fase de Análise e Planejamento. Na fase de Análise a aplicação legada é submetida ao ST Draco-PUC, que analisa a aplicação produzindo as Descrições XML do Modelo de Negócio da Aplicação Legada. As descrições são importadas na ferramenta Migration Project Planning (MPP), na qual o Engenheiro de *Software* vai planejar sua estratégia de migração, definindo as etapas do projeto de migração. O planejamento é armazenado no banco de dados do MPP e as Descrições XML do Planejamento da Estratégia de Migração da etapa a ser migrada são importadas no Apyon Studio para dar seqüência a fase de Migração. Na migração, o Apyon Studio gera a parte referente à tecnologia da aplicação, deixando-a totalmente funcional. O ST Draco-PUC pode ser usado para a conversão das regras de negócio da aplicação nas linguagens alvo. Na validação, o Engenheiro de *Software* define e aplica casos de teste para validar a migração.

Segue-se uma apresentação dos passos da abordagem AMGraA.

2.1. Avaliar Aplicação Legada

Esta fase preparatória é essencial para que o Engenheiro de *Software* possa entender o código da aplicação para conhecer suas necessidades e possibilidades relacionadas ao cenário de migração desejado e então apresentar opções e possíveis soluções para o problema.

O Engenheiro de *Software* avalia o código legado da aplicação, procurando respostas para questões, relacionadas: a estrutura do código legado; a possibilidade de alteração do código legado; ao armazenamento das informações da aplicação legada e as formas possíveis para integração. Ele terá o primeiro contato com a aplicação para conhecê-la, entender seu funcionamento e conhecer as possíveis formas de integração. Portanto, ao longo do processo de migração, conforme as necessidades da aplicação e requisitos de migração, o Engenheiro de *Software* especifica a forma de integração a ser utilizada para criar uma camada de comunicação com a parte legada da aplicação ou ainda com outras aplicações.

2.2. Construir Domínio e Transformador

Esta fase é essencial para a preparação do ambiente para a execução do processo. O Engenheiro de *Software* constrói o domínio, através de uma gramática para a linguagem dos programas, no ST Draco-PUC. Documentações e manuais sobre a linguagem do domínio e as regras sintáticas e semânticas da linguagem, orientam a edição da gramática.

Nesta fase o Engenheiro de *Software* edita as regras de transformações para a construção do transformador. O transformador construído é responsável por identificar e coletar descrições sobre o modelo de negócio da aplicação. Para orientar o Engenheiro de *Software* na identificação e coleta das descrições, foi definido um modelo para representar os elementos recuperados de uma aplicação legada como: Tabelas, seus Atributos, seus relacionamentos, as interfaces da aplicação, os atributos de interface, os procedimentos, as regras de negócio, os casos de uso, seus atores e sua origem.

As transformações identificam e extraem, do código da aplicação legada, dados sobre o modelo de negócio da aplicação. Os elementos recuperados são baseados nas

informações da gramática do domínio. Os dados coletados são armazenados como fatos na Base de Conhecimento (KB).

Na Tabela 1 são apresentados os padrões de reconhecimento que orientam na identificação e coleta das descrições do modelo de negócio da aplicação.

Tabela 1. Padrões de Reconhecimento para Coleta dos Dados

Elemento	Padrão de Reconhecimento
Tabelas e Campos	a) arquivos de dados: as tabelas são identificadas com base nos comandos de abertura e criação de arquivos de dados. Os campos são identificados com base nos campos destes arquivos; b) banco de dados: as tabelas são identificadas com base nos comandos de abertura e criação das tabelas de banco de dados. Os campos são identificados com base nos campos destas tabelas; c) estrutura de dados: as tabelas são identificadas com base nos comandos de definição de estruturas de dados. Os campos são identificados com base nos campos destas estruturas de dados.
Regras de Negócio	as regras de negócio são reconhecidas pela definição de unidades de programa, que pode ser: um programa, subprograma, procedimento, função ou bloco de comandos. Uma unidade de programa pode ser chamada por um evento de interface, por outra unidade de programa, ou pelos triggers, que são programas disparados por eventos de criação, alteração, exclusão, pesquisa e replicação do banco de dados.
Relacionamentos	os relacionamentos são reconhecidos pelos campos que formam um índice único em um determinado arquivo ou tabela A, e também são atributos de um outro arquivo ou tabela. Para arquivos ou tabelas que estão relacionados, mas não possuem os mesmos atributos, são também analisados os comandos de acesso aos arquivos ou banco de dados para reconhecer um possível relacionamento.
Hierarquia de Chamadas	a hierarquia de chamadas é definida pela seqüência de execução das unidades de programa do código legado.
Fluxo de Execução	o fluxo de execução de uma aplicação é identificado por comandos que determinam como será a execução da aplicação.
Casos de Uso	a) cenários da aplicação: cada opção do menu da aplicação dá origem a um caso de uso; b) scripts de linkedição: algumas aplicações legadas possuem scripts de linkedição que definem o relacionamento de vários programas fontes que darão origem a um único programa executável. Cada agrupamento deste dará origem a um caso de uso da aplicação; c) identificação manual: os casos de uso são determinados manualmente, por meio da hierarquia de chamadas do programas.
Interface e Objetos de Interface	a interface e os objetos de interface de uma determinada linguagem são identificados a partir dos comandos existentes na linguagem para definir a interface e seus objetos. Caso a linguagem não possua tais comandos, a identificação é realizada através de comandos que mostram informações na tela como, por exemplo, o comando DISPLAY.

Essas atividades são implementadas no transformador, que atua no código da aplicação legada, identificando e armazenando as informações na KB. Uma vez realizados os passos preparatórios, o Engenheiro de *Software* dá início a execução do processo de modernização da aplicação.

2.3. Analisar Aplicação Legada

Nessa fase inspeciona-se o código da aplicação legada, visando recuperar sua semântica, para obter os modelos de negócios da aplicação. Esses modelos contêm informações que expressam o negócio da empresa, independente da tecnologia e plataforma em que a aplicação foi implementada.

Nessa atividade, o Engenheiro de *Software* é auxiliado por um Sistema de Transformação (ST), que faz a análise sintática do código da aplicação, e coleta informações desse código. A Figura 2 mostra com mais detalhes essa atividade para obtenção das descrições do seu modelo de negócio.

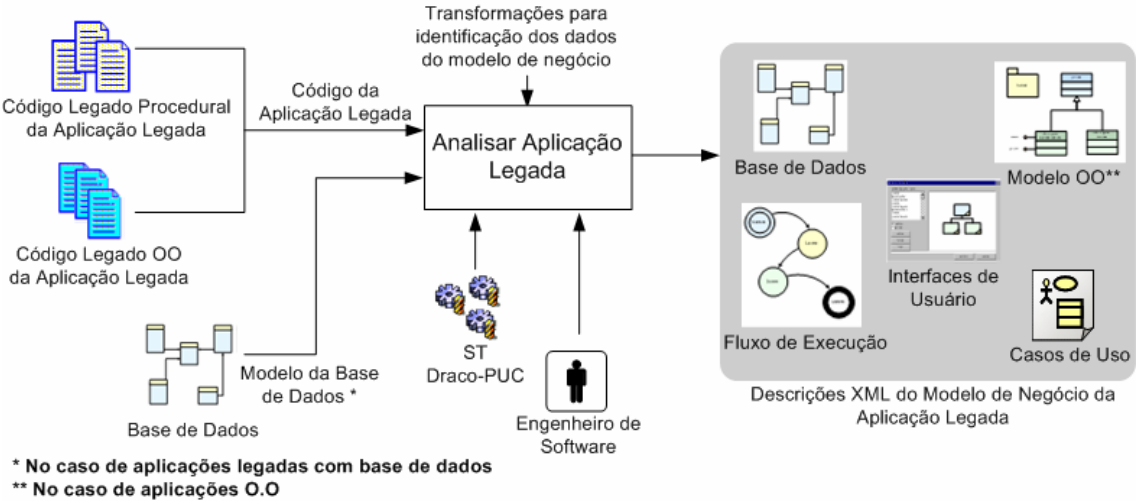


Figura 2 – Analisar Aplicação Legada

Conforme ilustrado, o Engenheiro de *Software* aplica os transformadores no Código Legado da Aplicação (Procedural ou OO) e na sua Base de Dados (ou sistema de arquivos, quando for o caso), para obter as Descrições XML do Modelo de Negócio da Aplicação Legada. As informações obtidas da análise são armazenadas em arquivos no formato XML, para uso das atividades posteriores da migração. Essas informações descrevem: Variáveis, Estruturas de Dados, Tabelas com seus campos e relacionamentos, Comandos de acesso a banco de dados ou arquivos, Fluxos de Execuções que determinam os casos de uso das aplicações, Interfaces com Usuário, Regras de Negócio, e outros recursos da linguagem utilizados pelas aplicações. No caso de aplicações Orientadas a Objetos (OO), obtêm-se ainda informações sobre encapsulamento, herança, composição, agregação, polimorfismo, conexões de mensagens, tipos genéricos e outros recursos das linguagens OO.

2.4. Planejar Projeto de Migração

Nessa fase, conforme ilustra a Figura 3, o Engenheiro de *Software*, parte das Descrições XML do Modelo de Negócio da Aplicação Legada para obter as Descrições XML do Planejamento da Estratégia de Migração (definindo “o que”, “quando” e “como” migrar). Para tal, inicialmente, importam-se as descrições XML obtidas na fase anterior através da ferramenta Migration Project Planning (MPP). Com auxílio do MPP o Engenheiro de *Software* pode Analisar inconsistências e duplicidades corrigindo inconsistências, redundâncias, ou quaisquer erros identificados nas descrições. Muitas

redundâncias ocorrem devido às várias manutenções sobrepostas nas aplicações legadas.

Em seguida, podem-se aprimorar as informações, re-especificando a aplicação, refinando as informações coletadas para atender novas necessidades de negócio, se for necessário. O planejamento inicia-se e o Engenheiro de *Software* então, pode: *Definir as etapas*, quando ele define as etapas do seu projeto de migração; *Cadastrar camadas de destino*, quando ele cadastrada as camadas que a nova aplicação terá; *Definir tecnologias para as camadas*, quando ele define as tecnologias que serão utilizadas em cada camada; *Especificar conjunto de dados a serem migrados*, quando ele especifica os dados a serem migrados a cada iteração do processo de migração; *Especificar comunicação*, quando por fim, ele especifica as formas de comunicação dos módulos migrados com a aplicação legada e com outras aplicações.

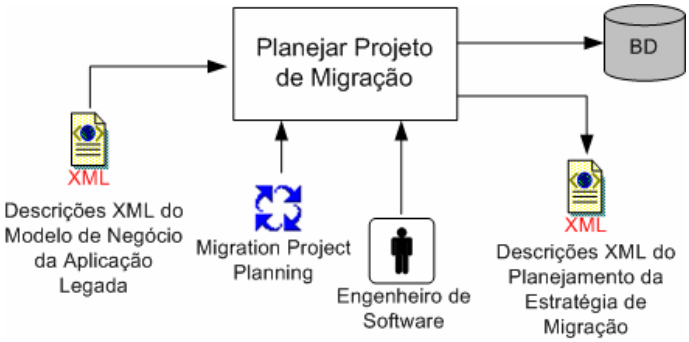


Figura 3 – Planejar Projeto de Migração

As Descrições do Planejamento da Estratégia de Migração também são persistidas em arquivo no formato XML e exportadas para o Apyon Studio, para serem utilizadas no próximo passo da abordagem.

A idéia é que a estratégia de planejamento dê flexibilidade para uma migração gradativa, considerando, além das particularidades de cada aplicação, os custos, recursos humanos, e outros recursos envolvidos.

2.5. Migrar Aplicação Legada

Essa atividade visa realizar gradativamente a migração da aplicação, conforme definido no planejamento. O Engenheiro de *Software* é apoiado pelas ferramentas Apyon Studio e o Sistema de Transformação (ST), conforme ilustra a Figura 4.

Normalmente, numa primeira etapa, migram-se somente os requisitos não-funcionais da aplicação (parte tecnológica), como acesso a banco de dados, suas interfaces e interações, e outros requisitos não funcionais, mantendo os requisitos funcionais, ou seja, as regras de negócios da aplicação. Posteriormente, migram-se os requisitos funcionais.

No Apyon Studio, o Engenheiro de *Software* pode *Reespecificar interfaces e componentes*. Por exemplo, podem-se reespecificar as interfaces e adicionar componentes que atendam a novos requisitos. Pode-se ainda *Gerar as interfaces de usuário*, *Gerar Controllers* necessários para a integração e comunicação com os demais módulos e com as regras de negócio obtidas da aplicação, e ainda *Gerar Documentação dos módulos migrados*, quando todas essas informações são adicionadas na documentação gerada pelo Apyon Studio.

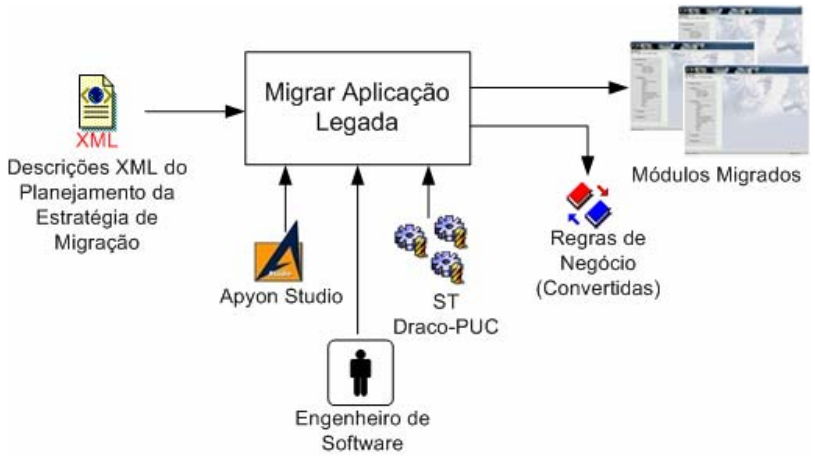


Figura 4 – Migrar Aplicação Legada

O Engenheiro de *Software* pode ainda realizar a migração das regras de negócio da aplicação. Esta atividade pode ser opcional, uma vez que o processo é gradativo e a migração das regras de negócio pode ser adiada, de acordo com o cenário da migração. O Engenheiro de *Software* pode migrar as regras de negócio com o auxílio do ST Draco-PUC ou pode reescrevê-las. É importante considerar que o código gerado para as regras de negócio pelo ST Draco-PUC deve ser analisado e trabalhado, para que a estrutura do código reflita a nova linguagem escolhida e não a linguagem da aplicação legada. Isto é importante para facilitar manutenções futuras no novo código da aplicação.

Esta fase oferece recursos que tornam mais flexível a migração da aplicação, uma vez que a aplicação vai ser migrada de forma gradativa e os requisitos definidos para um projeto de migração podem não ser iguais aos requisitos especificados para outros projetos. Desta forma, pode-se migrar somente requisitos não-funcionais da aplicação, como as interfaces, mantendo os requisitos funcionais, ou seja, as regras de negócio da aplicação. Assim, é preciso definir como será o acesso às informações legadas através das novas interfaces.

3. Estudo de Caso AMGraA

Um estudo de caso foi elaborado para testar a abordagem proposta. Trata-se de uma aplicação legada de gestão empresarial para indústrias, chamada Ômega. A Ômega tem 900 programas fontes, totalizando em torno de 315.000 linhas de código desenvolvidas em COBOL Microfocus 4.5, com 280 arquivos de dados (VSAM).

A Ômega possui os seguintes módulos: Básico: que contém a estrutura básica da aplicação e as informações comuns aos demais módulos; Financeiro: composto de Contas a Pagar, Contas a Receber, Fluxo de Caixa e Conta Corrente de Caixa e Bancos; Materiais: composto de Compras e Estoque; Vendas: composto de Pedidos, Faturamento, Cotas e Conta Corrente de Representantes; Produção: composto de PCP e Controle de Chão de Fábrica; Folha de Pagamento: composto de Folha de Pagamento, Ponto Eletrônico e Benefícios; e Contabilidade: composto de Controle de Patrimônio, Contabilidade e Livros Fiscais.

A empresa Orion-ASP, proprietária da Ômega, solicitou a migração para ser executada nas atuais plataformas de *hardware* e *software*. Esta solicitação foi devido aos problemas de manutenções na linguagem COBOL, que estavam cada vez mais

custosas, principalmente devido à escassez de profissionais com experiência em COBOL. Assim, os fatores motivadores para a modernização da Ômega eram utilizar uma linguagem de programação moderna e explorar os serviços e recursos da Internet. Desta forma, a empresa optou por migrar sua aplicação para a linguagem C# da plataforma ".NET".

Para realizar a migração da aplicação houve a colaboração de 4 participantes com os seguintes perfis: Participante 1: profissional com experiência em manutenções da aplicação legada; Participante 2: profissional com experiência em desenvolvimento de aplicações na linguagem COBOL; Participante 3: profissional com experiência em desenvolvimento de aplicações na plataforma .NET; Participante 4: Engenheiro de *Software*. É importante considerar também o conhecimento e a experiência do Participante 2 na utilização do ST Draco-PUC.

A aplicação Ômega não possuía documentação e, portanto, foram utilizados na realização da migração os programas fonte e o conhecimento e experiência do Participante 1 na aplicação legada. O desenvolvimento da migração ocorreu nas dependências do GOES com reuniões esporádicas com os Participantes 1, 2 e 3 em seus locais de trabalho e teve duração de 2 meses e 15 dias.

Segue-se uma apresentação da aplicação da abordagem para a reconstrução da aplicação Ômega.

3.1. Avaliar Aplicação Legada

Primeiramente, o Engenheiro de *Software* reuniu informações sobre a Ômega, no caso, somente o código fonte da aplicação, e avaliou-a com o propósito de conhecer suas funcionalidades. Foi constatado também que a aplicação Ômega estava particionada em módulos e que o código da aplicação era suscetível a alterações e, por isso, seria possível modificá-lo para que a aplicação legada pudesse ser chamada pelos novos módulos, à medida que a aplicação fosse migrada, através de um *wrapper* ou ainda um *Web Service*.

3.2. Construir Domínio e Transformador

Para dar suporte à abordagem AMGraA foi construído no ST Draco-PUC, o domínio COBOL, definido pela gramática livre de contexto, *parser*, *prettyprinter* e o transformador COBOL definido pela biblioteca de transformação COBOLKB, que é responsável pela identificação das informações sobre o modelo de negócio da aplicação legada. Para descrever as informações do modelo de negócio armazenadas na Base de Conhecimento (KB) em XML e permitir a integração entre as ferramentas utilizadas na abordagem, foi construído o transformador DracoKBXML.

A criação do domínio COBOL e do transformador de identificação levou cerca de 1500 horas, aproximadamente 2 meses, considerando os testes com a passagem dos programas. A criação deste domínio foi realizada pelo Participante 2, com a colaboração do Participante 1 e 4. É importante considerar também o conhecimento e a experiência do Participante 2 na utilização do ST Draco-PUC. O Participante 4 informava aos Participantes 1 e 2 as informações a serem recuperadas e a estrutura das descrições do modelo de negócio da aplicação a ser seguida e refletida no documento XML gerado.

O domínio COBOL criado reconhece aplicações COBOL escritas pelos fabricantes Micro Focus e Fujitsu. O transformador pode ser adaptado para atender a outras variações da linguagem COBOL. Isso implica em alterar o seu domínio e transformador, o que exigiria cerca de 500 horas, aproximadamente 21 dias. Este número é baseado no histórico de outros projetos de Reengenharia de *Software*.

Embora o transformador DracoKBXML tenha sido construído no contexto de um projeto para a modernização de uma aplicação COBOL, sua utilização é independente de linguagem, podendo ser utilizado futuramente em outros projetos.

3.3. Analisar Aplicação Legada

Nesta fase o Engenheiro de *Software*, faz a análise da aplicação legada, aplicando o transformador de identificação. Neste estudo de caso foram selecionados os módulos Básico, Vendas e Materiais para serem submetidos ao ST Draco-PUC, que aplicou as transformações construídas na fase anterior para a extração dos dados sobre o modelo de negócio da aplicação.

A Figura 5 ilustra a identificação das tabelas e dos campos do trecho de código COBOL à esquerda. À direita, tem-se a transformação *"t_file_description"*, responsável por reconhecer a seção de arquivos da *Data Division* do COBOL. Nesta seção têm-se as *File Descriptions* (FDs), que descrevem as estruturas que representam os arquivos ou tabelas usadas pela aplicação. Assim, em (1) é identificado o arquivo *"ARQEMP"* dando origem a um fato denominado "*table*". Para cada *table* têm-se um contador, um índice e o seu nome. Em (2) cada campo do arquivo é identificado dando origem ao fato "*field*", contendo o nome da tabela, um contador, o nome do campo, seu *label*, tipo de dado, formato, e outros atributos opcionais como o seu valor inicial, se é mandatário ou não, sua descrição, e seu *help*.

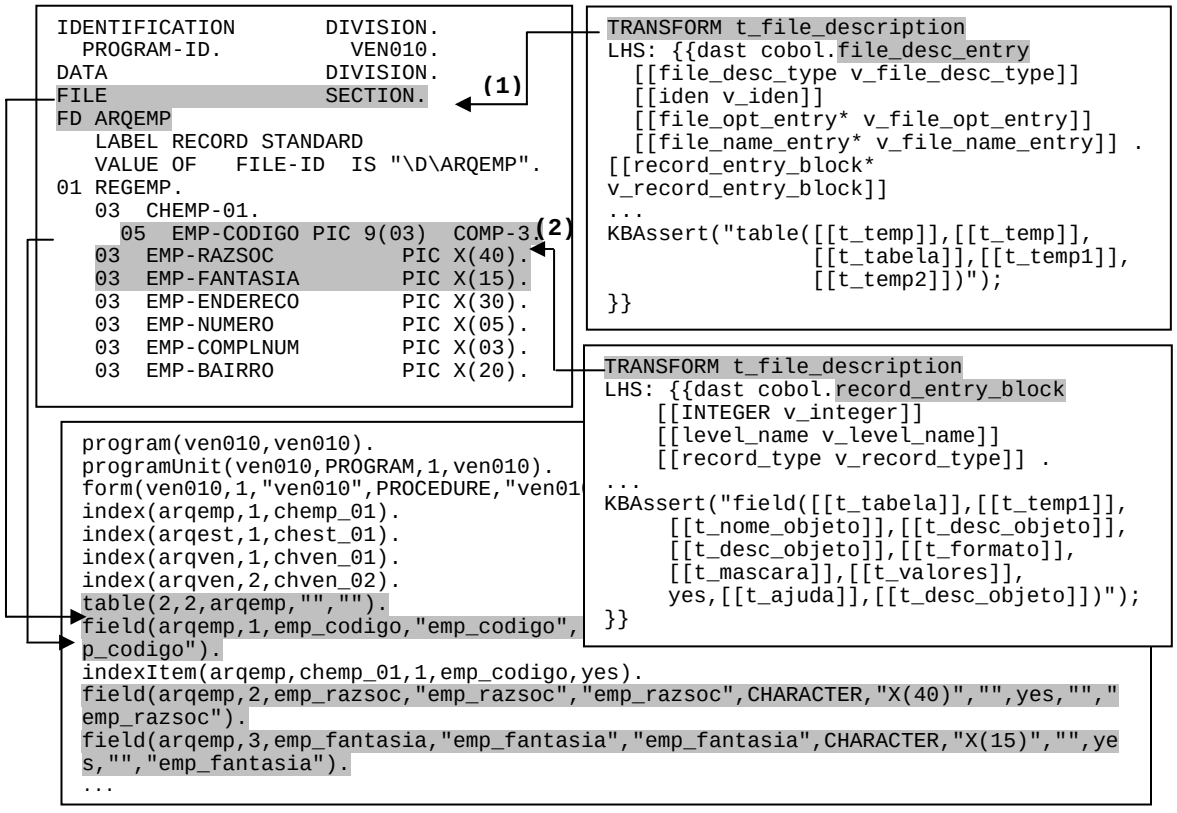


Figura 5 – Identificação da Tabela "ARQEMP" e seus Campos

Dos códigos legados são extraídos também informações sobre as interfaces de usuários, os objetos dessas interfaces, os fluxos de execução, relacionamentos entre tabelas e regras de negócio. Após a análise do código da aplicação, as informações do modelo de negócio, armazenadas na Base de Conhecimento (KB), são descritas no formato XML através da aplicação das transformações do transformador DracoKBXML.

A fase de Análise da Aplicação Legada levou cerca de 30 minutos, considerando a análise nos 3 módulos selecionados.

3.4. Planejar Projeto de Migração

Primeiramente, foi criado um projeto para a migração da aplicação, e em seguida importaram-se as descrições XML do modelo de negócio da aplicação legada, obtidas na fase anterior. Com os dados importados, pode-se analisar as informações coletadas pelo ST Draco-PUC, fazendo alterações como a modificação dos nomes de alguns programas e nome de campos e *label*s para nomes mais significativos. Em seguida, iniciou-se o planejamento do projeto de migração da aplicação Ômega, com ênfase para: (a) alterar aplicação para ser executada através de *wrappers*; (b) migrar Clientes e Vendedores e possibilitar o armazenamento em banco de dados; (c) migrar Pedidos para *Web* e (d) criar *web service* para Pedidos acessar a aplicação.

Após essa definição inicial, o Engenheiro de *Software* especificou a ordem em que essas etapas seriam executadas. Em seguida, foi feito o planejamento detalhado de cada etapa definindo os módulos e seus programas a serem migrados em cada etapa, a data do planejamento, o início e o fim da execução de cada etapa, como na Figura 6. À esquerda tem-se a definição da etapa *"Migrar Pedidos para Web"*, quando o programa *"ven030"* será migrado para a *Web* e à direita tem-se a especificação da exportação dos dados dessa etapa.

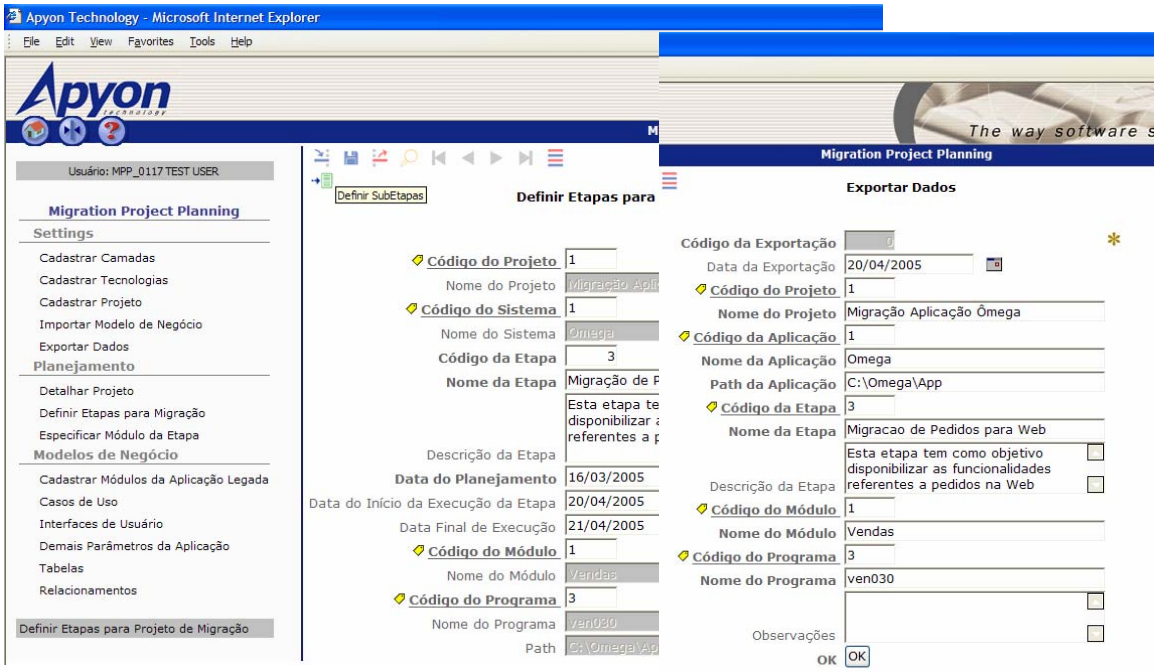


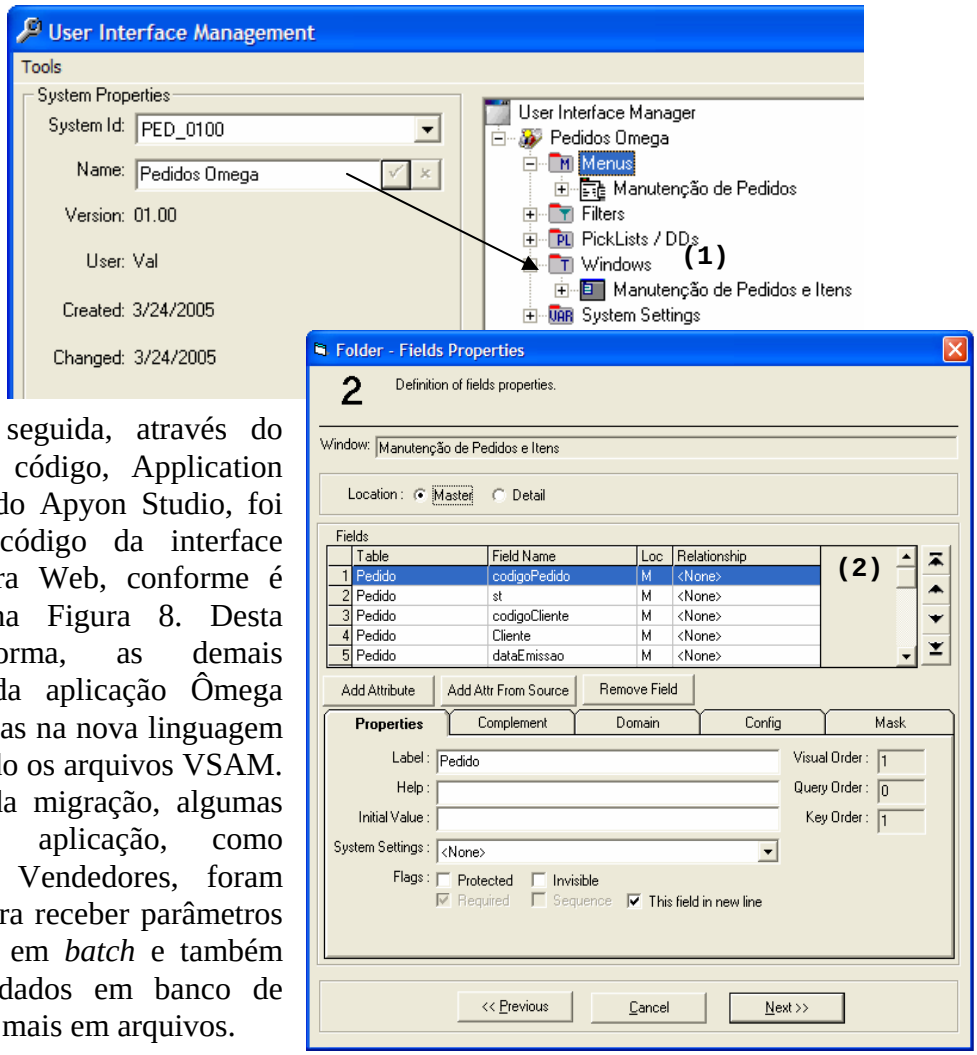
Figura 6 – Planejamento de Migração da Etapa "Migrar Pedidos para Web"

A exportação desse planejamento para a Apyon Studio é realizada através de um documento XML.

3.5. Migrar Aplicação Legada

Ao importar as descrições XML do modelo de negócio da aplicação e os requisitos do planejamento da migração da etapa no Apyon Studio, podem-se reespecificar as interfaces da aplicação, adicionando novas interfaces e componentes para a execução da aplicação, ou movendo ou alterando as interfaces e componentes já existentes.

Na Figura 7 são apresentadas as informações importadas, referentes à etapa "Migração de Pedidos para Web". No Apyon Studio, as interfaces identificadas podem ser visualizadas no item "Windows" (1) da treeview da janela "User Interface Manager". Conforme mostra a janela "Folder – Fields Properties" (2), a interface e os objetos de interface do programa "ven030" foram re-especificados. Foram alterados os nomes e labels da interface e de alguns objetos de interface.



Em seguida, através do gerador de código, Application Generator, do Apyon Studio, foi gerado o código da interface Pedidos para Web, conforme é mostrado na Figura 8. Desta mesma forma, as demais interfaces da aplicação Ômega foram geradas na nova linguagem C# acessando os arquivos VSAM. Ao longo da migração, algumas partes da aplicação, como Clientes e Vendedores, foram alteradas para receber parâmetros e funcionar em *batch* e também armazenar dados em banco de dados e não mais em arquivos.

Figura 7 - Descrições XML da Etapa "Migrar Pedidos para Web" importadas no Apyon Studio

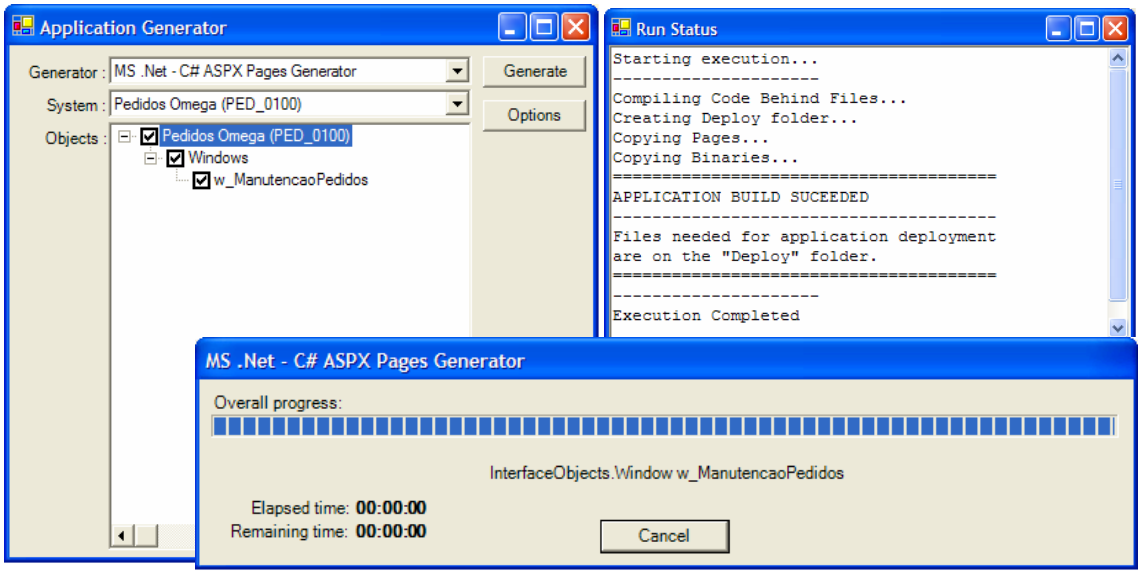


Figura 8 – Geração da Interface Pedidos em ASPX na Ferramenta Apyon Studio

A Figura 9 apresenta a nova interface para a manutenção de Pedidos para Web, gerada pelo Apyon Studio. Esta nova interface acessa os dados de Clientes e Vendedores, em COBOL, usando um web service implementado em C#. Esse *web service* foi construído para atender quaisquer aplicações da Ômega que necessitem acessar dados em COBOL.

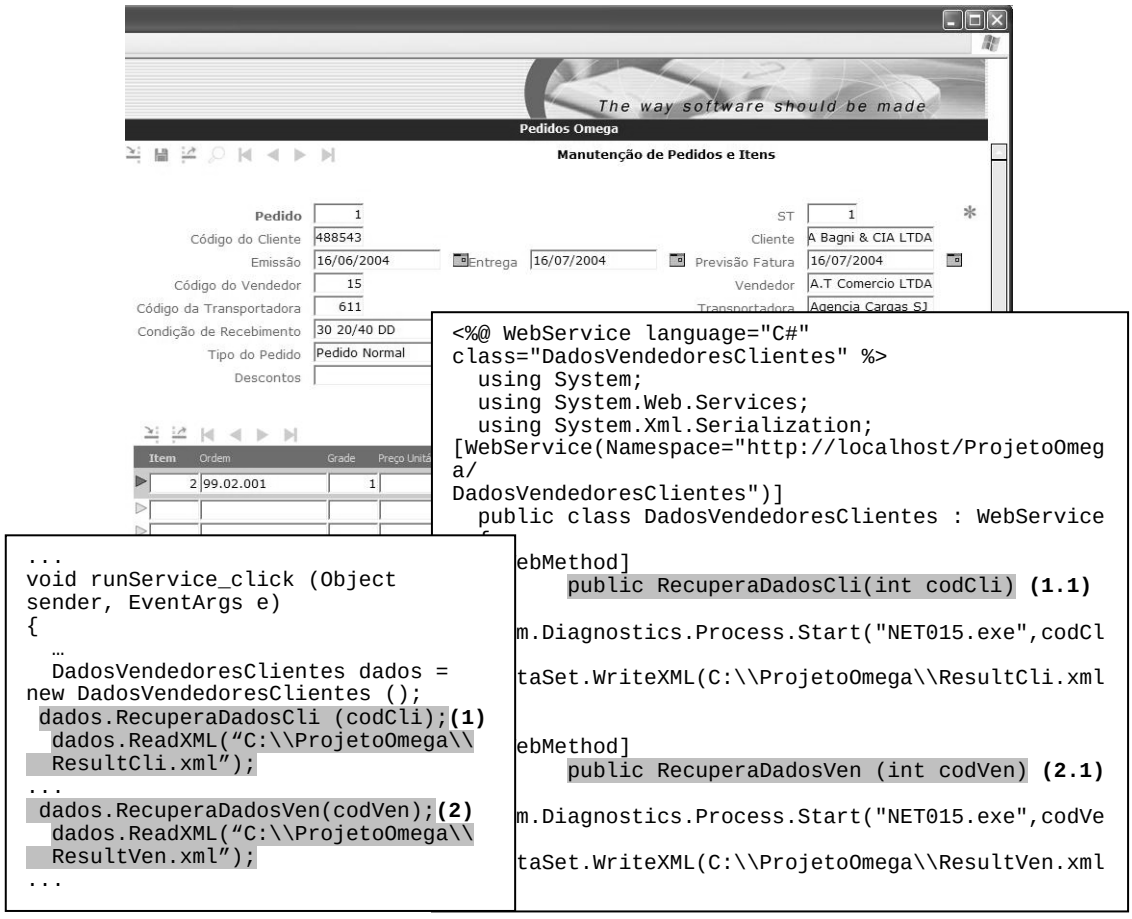


Figura 9 – Nova Interface Pedidos e a Chamada do Serviço do Web Service

Assim, a nova interface de Pedidos recupera dados de Clientes e Vendedores ao efetuar um novo pedido, chamando os métodos *"RecuperaDadosCli"* (1) e *"RecuperaDadosVen"* (2) do *web service*, passando como parâmetros o código de cliente e vendedor, respectivamente. O *web service* executa os métodos (1.1) e (2.1) e retorna os resultados em descrições XML, que são tratados pela aplicação. Desta forma, o usuário interage com o *web service*, ao invés do programa COBOL, que está sendo executado através de uma chamada externa.

Para que se tenha uma idéia da evolução, a Figura 10 mostra a execução da aplicação Ômega em C#, após a aplicação da abordagem AMGraA.

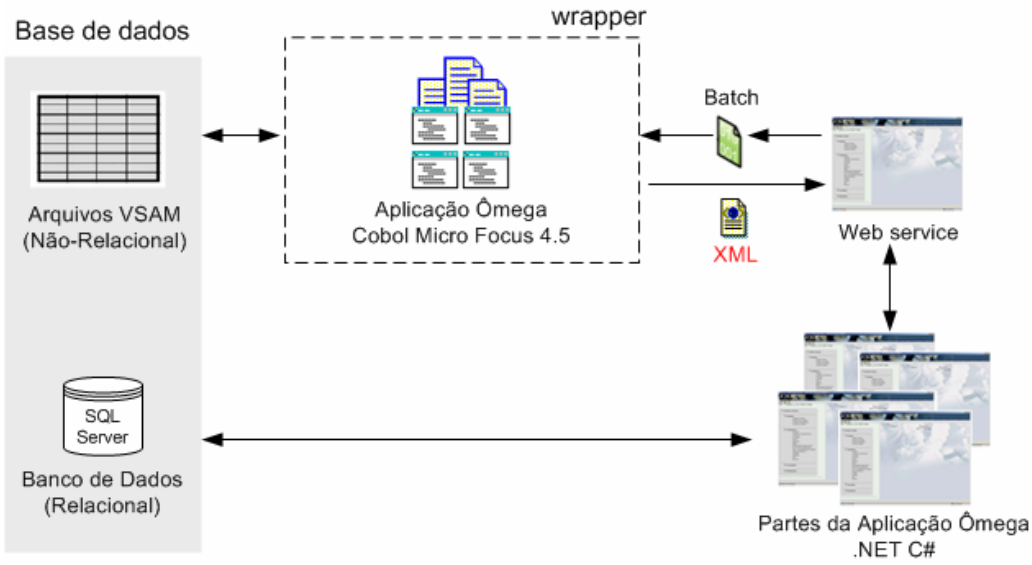


Figura 10 – Novo Modelo de Execução da Aplicação Ômega

Os resultados obtidos do projeto Ômega até o momento resultaram de três semanas de trabalho de um Engenheiro de *Software*, sem considerar o tempo gasto na escrita das transformações de *software* para a fase de análise e extração de dados do modelo de negócio.

4. Trabalhos Correlatos

Embora existam muitas abordagens para a migração de aplicações na literatura, ainda há uma carência por métodos que sejam viáveis para a aplicação na indústria, que ofereçam resultados práticos e que permitam o aproveitamento de investimentos anteriores em projetos de modernização realizados pela empresa.

A maior parte das abordagens tratam somente a mudança para o paradigma orientado a objetos, como em [Jacobson e Lindstrom 1991], [Markosian 1994], [Gall e Klösh 1994], [Penteado 1996], [Sneed 1996], [Riva 2000], [Fontanette 2002] e [Wilkening 1995], obtendo alguma representação em um nível mais alto de abstração. Em [Gall e Klösh 1994], os autores recuperam diagramas de fluxo de dados (DFD's), diagrama de entidade e relacionamento (DER) e um modelo orientado a objetos da aplicação. Já as abordagens mais recentes como em [Fontanette 2002], [Álvaro 2003] recuperam modelos de casos de uso e seqüência da aplicação.

Outra técnica bastante utilizada é o encapsulamento, como o proposto em [Gall and Klösh 1994]. O encapsulamento pode muitas vezes ser tão complexo e exigir um grande número de mudanças nos programas principais e periféricos, que é quase uma

reescrita da aplicação. O tempo e recursos gastos podem ser iguais, já que uma tecnologia nova também acompanha ferramentas novas de desenvolvimento rápido para agilizar o processo.

[Bisbal and Lawless 1999] falam de três tipos de migração - *"Cut-and-run"*, no qual é realizada uma parada total da aplicação antiga; "Phased", no qual a migração é realizada por partes e *"Parallel"*, no qual ambas as aplicações são mantidas. Para os autores a migração geralmente passa por um *"gateway"*, que possui um *"middleware"* para que a aplicação nova converse com a aplicação antiga, para que a aplicação antiga seja então migrada aos poucos. Esta é justamente a forma mais cara de se fazer e a que dá mais trabalho, pois uma aplicação que vai rodar com um *middleware* provavelmente terá que ser escrita novamente logo em seguida porque vai ficar "presa" ao *middleware*. Em alguns casos eles citam *"gateway"* apenas como uma infra-estrutura para permitir que as aplicações conversem, mas a programação tem que ser feita totalmente à mão. Na abordagem AMGraA, essas três opções podem ser usadas, sendo adotadas de acordo com o módulo a ser migrado.

5. Análise Crítica e Limitações

No decorrer da reconstrução da aplicação Ômega, foram identificados alguns pontos, são eles: **Recuperação de dados:** inicialmente foi proposta a recuperação dos trechos de código legado em que um elemento era recuperado e a sua visualização na ferramenta MPP. Mas devido ao ST Draco-PUC ler uma árvore sintática e não sequencialmente um arquivo texto, outros estudos serão necessários para realização desta tarefa. Esta atividade é um diferencial entre as abordagens de migração e, portanto, será considerada em trabalhos futuros; **Separação da tecnologia de negócio:** a abordagem AMGraA baseia-se na separação do negócio da aplicação da parte tecnológica (MDA). Esta separação permite uma migração gradativa das aplicações, reduzindo a complexidade envolvida nos processos de modernização de aplicações legadas e também no tempo, uma vez que os resultados são acompanhados gradativamente, conforme a migração da aplicação; **Biblioteca de transformação:** a biblioteca de transformação DracoKBXML construída no ST Draco-PUC para transformar os fatos coletados pelo ST Draco-PUC em formato XML, é independente de linguagem e, portanto, poderá ser reutilizada em qualquer domínio; **Comunicação entre módulos:** a abordagem proposta oferece a comunicação entre os módulos migrados e legados através de algumas tecnologias, mas o Apyon Studio ainda não oferece a geração automática delas como, por exemplo, dos *web services*.

6. Conclusões

Com as necessidades do mercado atual, e devido à evolução das linguagens de programação e o surgimento e amadurecimento de novas tecnologias, as aplicações precisam evoluir para acompanhar e atender a estas necessidades.

Cada vez exige-se mais que aplicações sejam integradas e distribuídas, seja para facilitar o contato com o cliente ou agilizar o processo da empresa. A partir dessa exigência, novas tecnologias surgiram para oferecer suporte a esse mercado. Esse quadro aprimorou-se com o amadurecimento das tecnologias para *Web*, sendo que a necessidade agora não só é converter a aplicação, mas adaptá-la essas novas tecnologias.

Motivados por estas idéias, este trabalho apresentou uma abordagem gradativa para a migração de aplicações legadas, que permite modernizar aplicações de forma gradativa, em conformidade com as prioridades da empresa e seus orçamentos e prazos, mantendo a integração entre as aplicações. A abordagem baseia-se na separação do negócio da aplicação da parte tecnológica. Esta separação permite uma migração gradativa das aplicações, reduzindo a complexidade envolvida nos processos de modernização de aplicações legadas e também no tempo, uma vez que os resultados são acompanhados gradativamente, conforme a migração da aplicação.

Para apoiar e facilitar os trabalhos do Engenheiro de *Software*, na execução da migração das aplicações, foram integrados um Sistema de Transformação ST, e as ferramentas Migration Project Planning (MPP) e Apyon Studio. O ST Draco-PUC e o Apyon Studio foram desenvolvidos isoladamente, mas foram complementares em suas funções no caso de migração de aplicações legadas. Com a integração dessas duas ferramentas, aliada a uma abordagem de migração gradativa, tem-se um ambiente para conversão e modernização de aplicações legadas.

É importante salientar que ambas as ferramentas foram desenvolvidas a partir de trabalhos acadêmicos, e que o Apyon Studio está disponível no mercado há mais de seis anos, enquanto que o ST recentemente vem sendo utilizado na indústria. A abordagem foi definida objetivando aproveitar os melhores recursos de cada uma dessas ferramentas, com foco em dois objetivos comuns: a migração de aplicações legadas e uma solução mais amigável de reconstrução e modernização de *software* para as empresas. Desta forma, estas poderão estudar a melhor forma de migrar suas aplicações em conformidade com seus objetivos e prioridades, para que esta tenha um baixo impacto, alto nível de produtividade, alta qualidade e baixos custos.

A grande inovação neste projeto é a flexibilidade do processo de migração, uma característica não comum nas abordagens e ferramentas disponíveis. A flexibilidade é imprescindível para se ter uma migração que se adapte a diversos ambientes, aplicações, tecnologias, arquiteturas, tamanhos de empresas, conhecimentos de profissionais, tempo, custo e volume de dados. Desta forma, a proposta é inovadora, produtiva e pode ser adaptada para outras combinações de ferramentas uma vez que a comunicação entre cada atividade é realizada através de descrições em XML.

References

SEACORD, R., PLAKOSH, D., LEWIS, A. G.. Modernizing Legacy Systems – *Software* Technologies, Engineering Processes, and Business Practices. SEI-Series in *Software* Engineering – Addison-Wesley, 2003. ISBN 0-321-11884-7.

BIANCHI, A.; CAIVANO, D.; VISAGGIO, G. Iterative Reengineering of Legacy Systems. IEEE Transactions on *Software* Engineering v.29, n.3, p. 225-241, March, 2003.

WILKENING, D. E. et al. A reuse approach to *software* reengineering. Journal of Systems and *Software*, v. 30, n. 1-2, p. 117–125, 1995. ISSN 0164-1212.

BISBAL, J.; LAWLESS D.; WU, B.; GRIMSON, J. Legacy Information Systems: Issues and Directions. IEEE *Software*, September/October 1999.

SYSTA, T. The relationships between static and dynamic models in reverse engineering java *software*. In: Proceedings of the 6th Working Conference on Reverse Engineering (WCRE'99). [S.l.]: IEEE Computer Society Press, 1999.

BIGGERSTAFF, T. J.; MITBANDER, B. G.; WEBSTER, D. E. Program understanding and the concept assignment problem. Communications of the ACM, ACM Press, v. 37, n. 5, p. 72–82, 1994. ISSN 0001-0782.

ALVARO, A. et al. Orion-RE: A Component-Based *Software* Reengineering Environment. In: Proceedings of the 10th Working Conference on Reverse Engineering (WCRE). [S.l.]: IEEE Computer Society Press, 2003.

PRICE, B.; BAECKER, R.; SMALL, I. A principled taxonomy of *software* visualization. Journal of Visual Languages and Computing, ACM Press, v. 4, n. 3, p. 211–266, 1993.

OLIVEIRA, A. L. C.. Uma Abordagem para Migração Gradativa de Aplicações Legadas. FAPESP/PIPE - Processo: 03/07851-4. Data inicio: 01/05/04.

FONTANETTE, V. Uma Abordagem para Migração Gradativa de Aplicações Legadas. Monografia de Qualificação. Departamento de Computação, UFSCar, 2004.

PRADO, A. F.;FONTANETTE, V.;GARCIA, V.C. et al. Reengenharia de *Software* usando Transformações (RST), Projeto CNPQ/RHAE - Nro: 610.069/01-2, Vigência: 08/2001 à 03/2004.Site: http://www.rst.dc.ufscar.br/.

FONTANETTE, V. et al. Reprojeto de Sistemas Legados Baseado em Componentes de *Software*. In: XXVIII Conferencia Latinoamericana de Informática (InfoUYclei), nov., 2002, Montevideo, Uruguai. Anais. Montevideo: Mastergraf SRL, 2002.

JACOBSON, I.; LINDSTROM, F. Reengineering of old systems to an object-oriented architecture. In: Proceedings of the Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'91). [S.l.]: ACM Press, 1991. p. 340–350. ISBN 0-201-55417-8.

MARKOSIAN, L. et al. Using an enabling technology to reengineer legacy systems. Communications of the ACM, ACM Press, v. 37, n. 5, p. 58–70, 1994. ISSN 0001-0782.

GALL, H.; KLÖSCH, R. Program transformation to enhance the reuse potential of procedural *software*. In: Proceeding of the ACM Symposium on Applied Computing (SAC'1994). [S.l.]: ACM Press, 1994. p. 99–104. ISBN 0-89791-647-6.

PENTEADO, R.D. Um Método para Engenharia Reversa Orientada a Objetos. São Carlos-SP, 1996. Tese de Doutorado. Universidade de São Paulo. 251p.

SNEED, H. M. Object-oriented cobol recycling. In: Proceedings of the 3rd Working Conference on Reverse Engineering (WCRE'96). [S.l.]: IEEE Computer Society Press, 1996. p. 169–178.

RIVA, C. Reverse architecting: an industrial experience report. In: Proceedings of the 7th Working Conference on Reverse Engineering (WCRE'2000). [S.l.]: IEEE Computer Society Press, 2000. p. 42–50.