CONSIDERAÇÕES SOBRE MIGRAÇÃO SEMI-AUTOMATIZADA

X

REESCRITADE APLICAÇÕES LEGADAS

Modernização semi-automatizada versus Reescrita (Fábrica de Software)

O Gartner, em seu documento intitulado “Code Transformation Solutions Can Provide a Viable Modernization Alternative”, G00218868 - Dale Vecchio - 2012-02-17, aborda diversos aspectos referentes às soluções de transformação/migração de códigos como alternativa altamente viável à modernização de aplicações legadas e frisa em suas principais conclusões que muitos fornecedores oferecem algum tipo de tecnologia de automação para apoiar a transformação/migração de uma aplicação, de uma linguagem ou de um banco de dados para outro. Algumas soluções proporcionam uma tradução apenas de sintaxe e outros fornecem uma transformação de arquitetura.

A entrega não apenas de código convertido, de conversão de sintaxe, mas de uma solução que realize a transformação da arquitetura, entregando uma arquitetura de classes orientadas a objeto prontas para ambientes .Net ou Java, é realmente a mais indicada. Sendo esse um dos pontos principais da proposta de valor do projeto 4Bears / XSEED de Modernização da aplicação > Modelagem de dados > Evolução da aplicação.

A principal missão é disponibilizar soluções e tecnologia que permitam aos clientes implementarem processos de transição das funcionalidades dos seus sistemas atuais para sistemas mais ágeis, revigorados tecnologicamente, de modo a melhorarem a sua competitividade nos mercados onde operam, devido a:

- Sistemas mais eficazes no suporte das suas operações
- Redução dos custos operacionais
- Reoxigenação das suas equipes
- Maior flexibilidade na evolução tecnológica e funcional dos sistemas

O ativo mais valioso do software legado é todo o vasto conhecimento funcional incorporado nesses sistemas, correspondendo a anos de evolução e aprimoramento das regras de negócio. Os ativos mais valiosos das novas tecnologias, pós-migração, são a sua grande flexibilidade, escalabilidade, seus atrativos visuais para a construção de interfaces mais amigáveis, e sua disponibilidade de desenvolvedores no mercado.

REESCRITA (OU REENGENHARIA) DE APLICACOES LEGADAS

Nos processos de reescrita (ou reengenharia) de aplicações legadas para Java ou .NET, estas são analisadas, reformadas, replanejadas e reprogramadas em uma nova plataforma tecnológica.

Vantagens

- As aplicações são modernizadas para espelhar o estado-da-arte em termos de tecnologia.
- As aplicações são modernizadas em termos das funcionalidades (processos de negócios) às quais são destinadas a implementar, procurando atender às necessidades atuais da comunidade usuária, alem daquelas que já atendia, enquanto legada.

Esforço

- Reescrever aplicações corporativas exige um esforço significativo, uma vez que requer um ciclo de vida completo de desenvolvimento para uma nova aplicação. Geralmente, projetos de reescrita levam vários anos, em torno de 4 a 5 anos, quando falamos em projetos em torno de 20mil a 25mil pontos de função.

Escopo indefinido

- Como sempre há um grande backlog de novas necessidades a implementar (processos de negócios) nessas aplicações legadas, os usuários, sabendo que está havendo um processo de reescrita (com nova análise funcional), vão exigir a implementação dessas novas funcionalidades, o que aumenta substancialmente a demanda de serviços nesse processo de substituição do aplicativo legado.

- Como migrações manuais, projetos de reescrita, são longos e caros, fica muito difícil resistir à adição de novas funcionalidades e requisitos durante o projeto. A tarefa de migração fica maior e mais arriscada, e obviamente fica muito mais complexo mensurar os resultados da entrega por completo, porque a nova aplicação já não faz o que o sistema legado fazia.
- Os novos recursos adicionados à aplicação, ao mesmo tempo em que a mesma é reescrita, irá definitivamente causar problemas para os usuários finais devido à sua exigência em aprender a usar as novas funcionalidades em pleno vôo. Usuários são extremamente resistentes a mudanças, as quais devem ser implementadas de forma estruturada. Um projeto de reescrita deve incluir os custos envolvidos com a formação e a perda de produtividade por causa da mudança de aplicação e novas funcionalidades.

Custos

- Os custos de reescrita, em sua grande maioria, pelas Fábricas de Software são mensurados baseados nos valores de pontos de função, que levam dúvidas pela complexidade da precificação com APF, como controle sobre produtividade, preços contingenciados, dependência de conhecimento técnico, possível renegociação de projeto. O que leva a custos elevadíssimos, com pouquíssima mitigação de risco.

Objetivos conflitantes

- Enquanto a equipe de desenvolvimento está tentando reescrever a aplicação legada, este sistema ainda está a pleno vapor, em produção. Ele deve evoluir para atender as necessidades das áreas de negócio, e assim as alterações são feitas continuamente. Tais mudanças são problemas para a equipe de migração, que agora tem de voltar atrás e refazer códigos já convertidos. Isso cria pressão vinda da equipe de migração para parar a evolução do sistema legado, que já não atende completamente a todas as áreas de negócio. Levando a atrasos significantes, o que resulta em maior orçamento necessário. Aumento significativo de riso de insucesso!

Incapacidade de mudanças significativas

- Durante a migração, os objetivos da empresa podem mudar, a equipe de migração vai aprender como lidar melhor com a reescrita, e os erros e acertos de migração serão assimilados em cada módulo. O problema é que o código convertido à mão contém as suposições que eram válidas nos módulos, com

os erros e acertos cometidos, no momento da conversão manual, e é doloroso para a equipe voltar atrás e mudar mesmo que parcialmente, estes módulos. Assim, o sistema migrado ou não se leva em conta as novas orientações, ou realmente se leva mais tempo para fazer entregar o projeto. Temos mais um fator de aumento significativo de risco de insucesso.

Qualidade de conversão irregular

- A migração realizada manualmente depende das habilidades individuais dos membros da equipe e de metodologias sólidas de projeto e desenvolvimento. Alguns possuem, naturalmente, desempenho melhor ou pior do que outros. A qualidade do código resultante irá variar consequentemente, elevando os custos de manutenção e prazos para o sistema convertido.

Conclusão:

- Alto custo
- Prazos extremamente extensos
- Alto risco de insucesso
- Dificuldade extrema em sincronizar as equipe de reescrita e manutenção do legado
- Sistemas bem ou mal estruturados / documentados dependendo das metodologias e formas individuais de programar adotadas

A Fábrica de Software consumirá a maior parte do tempo aprendendo e reescrevendo as regras de negócios do legado, o que depende do apoio direto dos recursos internos do cliente, em detrimento a especializar-se na entrega de melhor performance, entrega da evolução do legado.

Modernização semi-automatizada

A modernização de códigos semi-automatizada apresenta-se, desta forma, como a melhor abordagem em vários aspectos. Um planejamento consciente com um fornecedor experiente e maduro assegura alcançar os objetivos de orçamento e prazos previamente propostos. Baseado nas questões supramencionadas, a modernização de códigos semi-automatizada apresenta os seguintes aspectos:

Menor custo

- As modernizações de códigos semi-automatizadas acompanhadas de transformação de arquitetura, dependendo da complexidade e do número de linguagens envolvidas, possuem custos que podem variar de 30% a 50% menores do que um projeto de reescrita por fábricas de software.

Tempo mais reduzido

- Essas migrações normalmente levam entre 6 e 12 meses em projetos de baixa a média complexidade, podendo chegar a 3 ou 4 anos, dependendo da quantidade e complexidade das aplicações envolvidas, integrações entre várias aplicações diferentes, quantidade de linhas de códigos, etc.

Escopo definido

- A modernizações de código semi-automatizada não adiciona novas funcionalidade durante o processo de migração, com exceção de alterações ocorridas no sistema legado, por razões de "caráter legal ou críticas para a continuidade do negócio", previamente negociadas com o gerente de projeto da migração. Evitam-se, desta forma, problemas vistos nos processos de reescrita com evolução em pleno voo de modernização de códigos apontados acima. A possibilidade de paralelismo de trabalho entre as equipes que migram e as que evoluem, permitem que um determinado módulo já migrado e implementado na nova arquitetura possa receber, logo em seguida, o trabalho da equipe de evolução, aí sim, trabalhando-se com um escopo e prazos bem definidos, podendo aplicar um trabalho de refatoração do código, melhorando a performance e evoluindo o que for previamente mapeado junto ao cliente. Essa evolução pode incluir a modelagem e modernização de base de dados, antes da refatoração de código.

Sem metas conflitantes

- A equipe de manutenção de software legado pode até adicionar novas funcionalidades no sistema em produção durante a migração, dentro das exceções anteriormente citadas e em caráter excepcional, e a solução de modernização de código irá converter novamente aquele módulo, o que será considerada uma re-migração, mantendo-se assim o fluxo de trabalho fluindo.

Mudanças de curso são mais fáceis:

- A modernização de códigos semi-automatizada é baseada em um conjunto de regras de migração já construídas e implementadas nas ferramentas de conversão, que são aplicadas a todos os módulos de um sistema legado durante processo de migração até suas entrega. Caso haja necessidade de novos rumos tecnológicos, essas regras e ferramentas podem ser ajustadas para atender novas demandas.

Qualidade na conversão

- As regras de modernização dos códigos usadas pelos *parsers* (filtros conversores) têm o efeito de impor um "estilo" de codificação e arquitetura consistente em todos os sistemas. Essas regras podem ser ajustadas para mudar o estilo. Algumas regras de migração se concentram em converter os conceitos da linguagem com precisão; Isto assegura uma tradução confiável. Algumas regras de migração se concentram em "limpar" construções resultantes inábeis, garantindo um código limpo, sem duplicidade, sem códigos “mortos” que já não são sequer usados.

Conclusão

- Certeza de conclusão com sucesso do projeto de migração
- Menores custos
- Prazos menores – cerca de 1/3 terço dos projetos de reescrita.
- Baixo risco
- Elevada escalabilidade para novas funcionalidades

Associado à modernização (transformação) de códigos semi-automatizada, essa abordagem permite que se possa investir mais tempo e orçamento na evolução da aplicação, em melhorias de performance, em atender demandas reprimidas dos vários setores de negócio. Desta forma, atender ao ciclo completo de Modernização > Modelagem de dados > Performance da aplicação > Evolução, com menos tempo e budget necessários a investimento.

Além dos benefícios já citados proporcionados pela modernização semi-automatizada de aplicações legadas, pode-se destacar:

Performance

- Estatísticas de operações de banco
- Estatísticas de operações com arquivos
- Estatísticas de execução de programas
- Manutenção do sistema em andamento, sem descontinuidade

Documentação

- Código legado fica comentado por completo no código migrado
- Documentação de fontes x base de dados
- Documentação de programas vinculados x dependências
- Documentação que facilita na gestão de testes e entrega de artefatos
- Documentação completa do sistema migrado

Padronização

- Conversor aplica metodologia padrão para que todos os fontes tenham a mesma característica, hierarquias e heranças;
- Maturação de um Framework próprio de desenvolvimento e produção;
- Estrutura de classes Java pronta para processos BPM ou WebService

CONSIDERAÇÕES SOBRE FORMS / REPORTS E JAVA

Principais características de Forms / Reports:

1) **Produtividade**

O ambiente Forms / Reports proporciona uma produtividade bastante elevada no que diz respeito ao desenvolvimento e manutenção de aplicações, o que é uma grande vantagem em se tratando de grandes aplicações, como costumam ser as aplicações corporativas.

2) **Disponibilidade de recursos humanos para desenvolvimento / manutenção**

O ambiente Forms / Reports enfrenta uma escassez de recursos de desenvolvedores, por ser uma tecnologia ultrapassada e não comum no meio universitário de formação desse tipo de recurso. Desta forma, este é um aspecto bastante negativo dessa plataforma.

Associado a essa baixa disponibilidade de recursos de desenvolvedores, normalmente os desenvolvedores atuais, existentes nas empresas, além de escassos, estão envelhecendo, caminhando para aposentadoria, o que traz problemas de continuidade futura para a evolução / manutenção das aplicações nessa tecnologia.

3) **Aplicações atadas às ferramentas de software Oracle**

O ambiente Forms / Reports é totalmente proprietário (fechado) em torno das tecnologias Oracle, exigindo que a aplicação use somente as ferramentas de Application Server e Base de Dados Oracle, não permitindo, desta forma, qualquer flexibilidade quanto à seleção de outras ferramentas com menores custos e maiores potenciais de performance.

4) Aplicações para plataforma WEB / Mobile

A forma como as aplicações geradas com Forms / Reports se comportam na WEB é quase como se fosse uma arquitetura Client / Server, com suporte WEB, implementada sobre *Applets* Java bastante pesados; o que implica em problemas de performance, na necessidade de estações com alta capacidade de processamento e dificulta o uso dessas aplicações em plataformas de mobilidade (smartphones, tablets, etc.). Há relatos de vários problemas de incompatibilidades com Browsers, por conta dessa implementação de Applets.

As aplicações Forms / Reports necessitam serem complementadas com aplicações em Java para plataforma móvel, exigindo dois tipos de perfis técnicos na área de desenvolvimento: uma equipe legada em Forms / Reports e outra em Java. Desta forma, dificultando a evolução e novos desenvolvimentos de aplicações, bem como a padronização de recursos humanos e padrões de desenvolvimento (metodologias / ferramentas) nesse tipo de ambiente.

5) Evolução da tecnologia

Apesar de ter o suporte assegurado para essa tecnologia Forms / Reports pela Oracle, trata-se de uma tecnologia legada, com dificuldades de evoluir na direção das novas tecnologias que vão surgindo no ambiente de TI – mobilidade, nuvem, bigdata...

Esse fato, associado à escassez de recursos humanos, dificulta a evolução das aplicações corporativas que estejam nessa tecnologia.

Principais características da tecnologia Java:

1) Produtividade

O ambiente tradicional de desenvolvimento de aplicações Java apresenta uma baixa produtividade tanto na criação quanto na manutenção de aplicações, tornando-se uma desvantagem significativa em se tratando de grandes aplicações, como costumam ser as aplicações corporativas.

Contudo, a abordagem de ferramentas Java adequadas à migração das aplicações Forms / Reports e adaptadas para tal (Framework e IDE), proporcionam produtividade de desenvolvimento / manutenção de aplicações em níveis similares aos observados nessa plataforma original, permitindo uma solução de continuidade para a equipe legada Forms / Reports.

2) Disponibilidade de recursos humanos para desenvolvimento / manutenção

A plataforma de desenvolvimento Java apresenta uma quantidade de recursos humanos de desenvolvedores muito mais ampla, por conta de ser esta a plataforma mais comum adotada no meio universitário, o que facilita a contratação e reposição de desenvolvedores.

Associado a esse fato, a abordagem adotada com ferramentas adequadas de migração Forms / Reports para Java, proporciona um ambiente de desenvolvimento (IDE) com muita similaridade ao ambiente Forms / Reports, permitindo que os desenvolvedores legados possam ter continuidade de trabalho, com certa tranquilidade, nessa nova plataforma Java, com baixa necessidade de treinamento, dada às características do IDE Java adaptado a essa realidade.

3) Aplicações não vinculadas a qualquer plataforma de hardware / software

As aplicações desenvolvidas na plataforma Java são independentes de hardware e software, podendo ser executadas em quaisquer plataformas homologadas para processamento Java. Com isso assegurando total liberdade na escolha de software Application Server, bem como de Banco de Dados.

4) Aplicações para plataforma WEB / Mobile

As aplicações desenvolvidas em Java são naturalmente habilitadas para plataforma WEB e, com alguns ajustes, para a plataforma WEB *mobile*. Desta forma, facilitando o *deployment* de aplicações para essas plataformas.

Alem disso, como a tecnologia de desenvolvimento é comum, caso sejam necessárias aplicações *mobile* para serem executadas diretamente nesses dispositivos, a equipe Java facilmente pode adquirir conhecimentos para o desenvolvimento de aplicações voltadas para essa plataforma com o Java *mobile* (Java Android).

5) Evolução da tecnologia

A tecnologia Java, ao lado da Dot.Net, é a mais moderna, em termos de aplicações corporativas, e a que mais evolui, por conta dos elevados investimentos que são realizados pelas empresas lideres do mercado de TI, bem como por toda a comunidade mundial de desenvolvedores.

Esse fato assegura uma evolução permanente das aplicações corporativas baseadas nessa tecnologia.

E na forma como as boas tecnologias de migração de Forms / Reports para Java operam (usando uma camada de classes que emulam os comandos de programação e comportamento do Forms / Reports, e todos os conceitos de orientação a objeto proporcionados pela tecnologia Java) a evolução das aplicações migradas é assegurada sem maiores esforços para a equipe de desenvolvimento da empresa, uma vez que as atualizações evolutivas são implementadas nas ferramentas IDE e Framework usadas no processo de migração.

6) Portabilidade

Sistemas desenvolvidos em Java podem ser executados em quaisquer servidores que tenham um application Server Java, inclusive os mainframes série Z da IBM. Isto permite uma grande flexibilidade na hora de se efetuar aquisições para expansões / trocas.

7) Escalabilidade

Os sistemas desenvolvidos em Java são totalmente escaláveis, pois podem ter suas execuções distribuídas em múltiplos servidores de aplicação, permitindo elevada escalabilidade horizontal.

8) Nuvem

Aplicações Java são facilmente publicadas na Nuvem, permitindo uma evolução técnica e de arquitetura sem maiores dificuldades.

9) Segurança

O ambiente de execução Java é um dos mais seguros na plataforma Open.

10) Desempenho

O desempenho das aplicações Java chegam muito próximo àquele encontrado nas aplicações escritas em C, as quais são as de maiores performance na plataforma Open.

11) Flexibilidade

Aplicações Java, desenvolvidas com os padrões de orientação a objeto (OO) são extremamente flexíveis e aproveitam ao máximo o conceito de reusabilidade, facilitando as manutenções desses sistemas, assim como aumentando a produtividade no desenvolvimento / manutenção dos mesmos.

12) Longevidade

A tecnologia Java aparenta ser a que tem maior potencial de longevidade das tecnologias atuais para a plataforma Open. A arquitetura J2EE (Java enterprise edition) foi criada no ano de 2.000 e vem sofrendo constantes implementações ao longo desses anos, acompanhando a evolução das tecnologias de informática e comunicações. É suportada pela maioria da comunidade dos fabricantes de hardware e software, o que assegura um comprometimento com o futuro dessa plataforma.

Sistemas corporativos são feitos para durar décadas e a tecnologia Java parece ser a mais indicada para abrigar novos desenvolvimentos de sistema visando o longo prazo.

13) Suporte

A tecnologia Java é suportada pelos maiores fabricantes de hardware / software, incluindo-se ai: IBM, Oracle, HP, Unisys, Dell.

14) Pessoal técnico

Os profissionais com formação Java são abundantes no mercado, uma vez que a maioria das faculdades de TI tem a linguagem Java como padrão acadêmico.