Migração WebISS MainFrame

Introdução

Muitos dos desenvolvedores da plataforma .NET desconhecem o fato de que é possível rodar aplicativos desenvolvidos com Visual Studio em plataformas não Windows. Muitas vezes, aplicativos desenvolvidos para o framework .NET que precisam rodar em outras plataformas são reescritos em outras linguagens, o que causa impacto em custos e retrabalho desnecessário. Entretanto, através de um projeto de software livre chamado Mono, é possível executar aplicativos .NET com baixo custo e pouco esforço. Através do Mono é possível rodar aplicativos escritos em C# em Linux, MacOS, BSD , iPhone, entre outras plataformas.

Atualmente é muito comum as empresas trabalharem com diversos tipos de plataformas de hardware e software. Por isso é importante conseguir que todo esse conjunto de tecnologias sejam capazes de conversar entre si trocando dados e informações.

Por isso a Microsoft tem desenvolvido diversas frentes de trabalhos para reforçar a questão da interoperabilidade de seus produtos com de terceiros. Uma das primeiras iniciativas nesse sentido foi a padronização do núcleo da tecnologia .NET (CLR e Linguagem C#) que resultaram em algumas especificações ECMA – European Computer Manufacturers Association – tais como o ECMA 334 e o ECMA 335. Juntas, estas duas especificações descrevem todo o funcionamento da tecnologia .NET e isto que possibilita a qualquer empresa criar sua própria versão do .NET, independentemente do sistema operacional ou plataforma de hardware. Foi justamente esta iniciativa da Microsoft em padronizar a linguagem que possibilitou que a comunidade de software livre pudesse criar um clone do .NET, o Projeto Mono. www.mono-project.com

Histórico do Mono

Em 2001, quando a Microsoft padronizou as especificações do .NET, Miguel de Icaza trabalhava como líder do projeto GNOME, uma das interfaces gráficas mais importantes para desktop Linux. Miguel buscava uma plataforma capaz de atender algumas necessidades de sua empresa e encontrou na plataforma .NET todas as qualidades que procurava. Segundo Miguel de Icaza, *"Durante muito tempo programei em C, mas sempre estava procurando uma linguagem melhor. Não havia nenhuma favorita, pois eu sentia que havia algo incompleto. O Java estava muito próximo. O problema é que era a linguagem de programação com melhores recursos, mas a Sun parou de ouvir os usuários. A receita da Microsoft foi simples: pegar o Java e fazer todas as alterações que a Sun tinha ignorado. O Java era bom, mas não tanto quanto poderia ter sido. O C# é simplesmente meu sistema ideal de programação".*

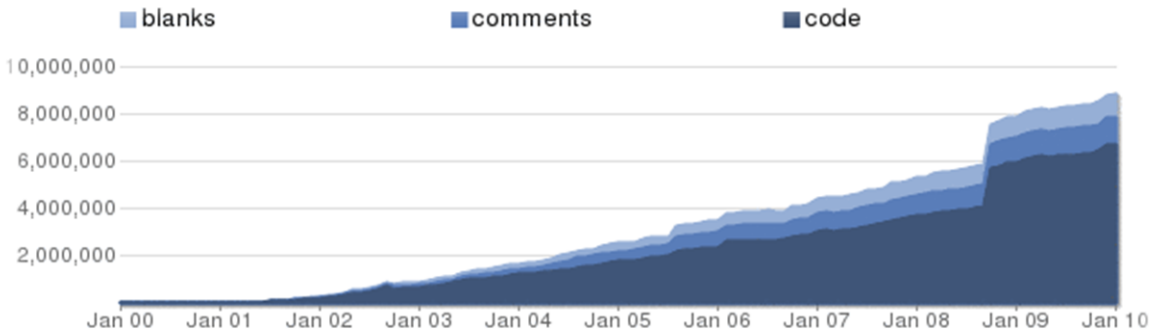

Miguel de Icaza

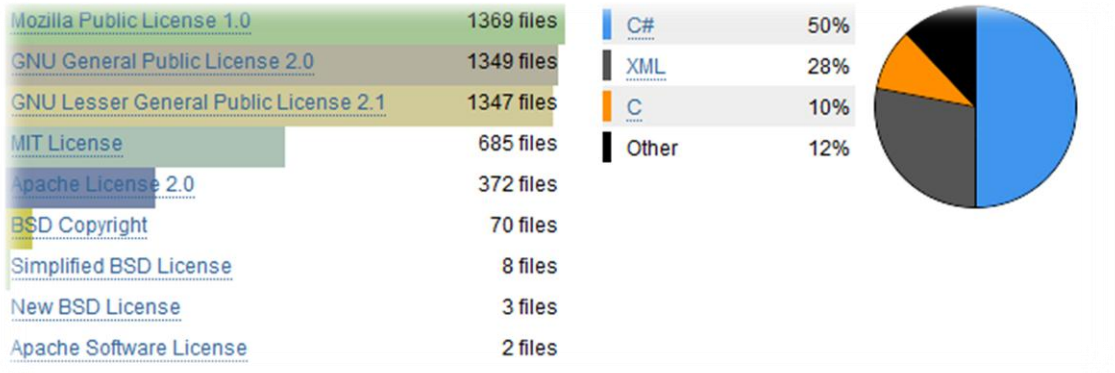
Com esta visão ele decidiu criar o Projeto Mono. Esta iniciativa seria uma implementação open source do .NET. Segundo o próprio Miguel "*o Mono é livre: uma fundação sólida voltada à inovação. Eu tenho os mesmos direitos que qualquer pessoa. Posso desenvolver algo em cima do que os outros desenvolveram. Quando se fala no Mono, é exatamente a mesma coisa. Estamos tentando trazer todas as coisas boas que a Microsoft fez com o .NET*". Em 2009 a Microsoft concedeu ao Miguel o título de MVP em C#.

Com a ajuda de apenas alguns programadores Miguel iniciou o Projeto Mono em sua própria empresa, a Ximian. O ponto de partida foi o desenvolvimento de uma runtime, ou uma máquina virtual responsável por converter o código IL (Linguagem intermediária) em código de máquina nativo. Simultaneamente foi

desenvolvido um compilador C# open source Mono CSHARP Compiler, o MCS, e a escrita das bibliotecas do .NET framework como open source. Não demorou muito para que a iniciativa de Miguel se tornasse popular a ponto de sua empresa ser comprada pela Novell. Atualmente, a Novell comercializa produtos baseados no Mono. Existem várias implementações do compilador C# disponíveis: o MCS, compilador para runtime 1.1; o GMCS, compilador para runtime 2.0; o SMCS, compilador para runtime 2.1 e para aplicações Moonlight, além do lançamento mais recente, o DMCS que foi iniciado com Mono 2.6 para C# 4.0. O suporte ao C# 4.0 está disponível na versão de desenvolvimento do Mono e pode ser baixado do site oficial ou diretamente dos repositórios de código do projeto.

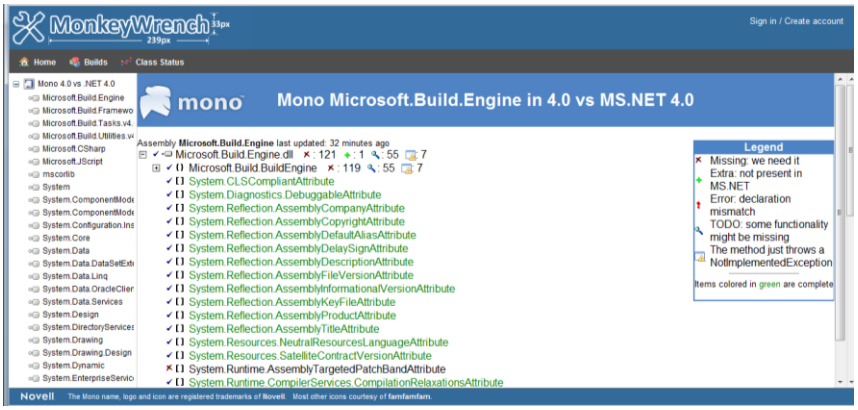
Atualmente o Projeto Mono atingiu a marca de seis milhões de linhas de código. Deste total 50% são escritas em C# e o restante em outras linguagens tais como XML, C, Javascript, Visual Basic, C++, SQL, HTML e Shell Script. Para testar essa grande quantidade de código há vários servidores em diferentes plataformas operando 24 horas por dia compilando e rodando testes. Diariamente são gerados relatórios de problemas os quais são automaticamente enviados aos desenvolvedores. Também é possível consultar esses relatórios pela Internet. Quando uma aplicação .NET está sendo portada para um outro sistema e a equipe se depara com um problema é possível consultar esses relatórios para verificar se existe algum problema na funcionalidade que a aplicação esteja usando e corrigi-la.





Compatibilidade

Podemos ver abaixo o status de implantação do Projeto Mono correspondente a cada versão do .NET. Em tempo real é possível acompanhar pelo site http://wrench.mono-project.com com uma análise do código fonte do Mono frente à implementação da Microsoft:



Legenda:

Implementado	Parcialmente Implementado	Não Implementado

.NET 4.0

- C# 4.0
- ASP.Net 4.0
- ASP.Net MVC 1 and MVC 2
- System.Numerics
- Managed Extensibily Framework – *Contribuição da Microsoft*
- Dynamic Language Runtime - *Contribuição da Microsoft*
- Client side OData - *Contribuição da Microsoft*
- Parallel Framework and PLINQ
- CodeContracts - *API complete, partial tooling*
- EntityFrameworks – *não disponível*
- Server-side OData – *depende do Entity Framework.*

.NET 3.5

- C# 3.0
- System.Core
- LINQ
- ASP.Net 3.5
- ASP.Net MVC
- LINQ to SQL -

.NET 3.0

- WCF - *silverlight 2.0 subset completed*
- WPF – *sem planos para implementação*
- WF – *Será implementado o WF 4 em futuras versões do Mono.*

.NET 2.0

- C# 2.0 (generics)
- Core Libraries 2.0: mscorlib, System, System.Xml
- ASP.Net 2.0 - *except WebParts*
- ADO.Net 2.0
- Winforms/System.Drawing 2.0 -

.NET 1.1

- C# 1.0
- Core Libraries 1.1: mscorlib, System, System.Xml
- ASP.Net 1.1
- ADO.Net 1.1
- Winforms/System.Drawing 1.1
- System.Transactions
- System.Management - *does not map to Linux*
- System.EnterpriseServices - *deprecated*

Sun x Microsoft

A disputa entre e Sun Microsystems e a Microsoft é antiga. Afinal, são empresas que pensam e agem de forma distinta. A Sun era uma empresa de hardware. Sua fonte de receitas era focada fortemente na venda de hardware. Entretanto, a empresa sabia que para vender seus produtos era necessário um software para seus equipamentos. Esse foi o motivo que levou a Sun a criar a plataforma JAVA de desenvolvimento de software. Através do JAVA a Sun oferecia recursos que seus clientes buscavam, tais como:

- Segurança, padronização e desempenho;
- Garantia de funcionamento e compatibilidade do software com o hardware;
- Mão de obra abundante;
- Uma plataforma de software consistente e robusta;
- Uma grande comunidade de desenvolvedores open source;
- Apoio de grandes empresas (inclusive de concorrentes, como a IBM);
- Aceitação do JAVA como uma linguagem empresarial.

O mais incrível dessa história é que, apesar de ter atingido todas essas difíceis metas, a Sun acabou se envolvendo em uma crise financeira, que culminou na venda para a Oracle, uma empresa de software.

Qual teria sido o erro da Sun se o hardware dela era um dos melhores e tinha uma plataforma de desenvolvimento de sucesso no mercado empresarial? Foram vários fatores, desde a má gestão de recursos, deficiência em enfrentar a concorrência com produtos mais baratos até o envolvimento de interesses pessoais. Mas um aspecto se destaca: Uma empresa de tecnologia não pode parar de ouvir seus desenvolvedores. A Sun criou um órgão praticamente estatal que concentrava o poder de deliberar e definir os rumos da plataforma JAVA, o JCP – Java Community Process.

Em 1998 a Sun tinha uma tecnologia de software 10 anos à frente de qualquer concorrente. Mas o foco não estava no software. Para a diretoria o software não tinha valor agregado. A empresa precisava vender hardware para as empresas e não pacotes de software para os desenvolvedores. Um exemplo muito claro disso foi a compra da StarAdvison , empresa alemã produtora de uma suíte de escritório concorrente do pacote Office da Microsoft. Ao adquirir a StarAdvison a Sun abriu o código fonte da suíte e a nomeou OpenOffice. Um pacote poderoso de aplicativos para escritório e compatível com todas as plataformas disponíveis no mercado, este passa a ser distribuído gratuitamente com o único objetivo o de confrontar o pacote Office da Microsoft.

Mas por que uma empresa de hardware compraria uma suíte office e o ofereceria de graça para todo mundo usar? Certamente não foi por simpatia com o movimento de software livre. Afinal, cada instalação de OpenOffice significa um pacote Office a menos no mercado. Consequentemente, menos domínio da Microsoft no mercado.

Nascimento e popularização do C#

Com o avanço do Java no mercado corporativo a Microsoft criou sua própria versão de JAVA, o J++, que acabou em um processo da Sun contra a Microsoft por quebra de patente. Depois de muitas brigas e audiências em tribunais, a Microsoft decidiu que era hora de mudar e criar algo novo, realmente à altura da Plataforma Java ou até mesmo melhor. Surge em 2001 a Linguagem C# e a plataforma .NET.

Diferentemente da Sun, a Microsoft registrou o C# e toda a infraestrutura básica da plataforma .NET como aberta. Uma norma gerenciada por uma instituição que regula padrões da indústria Europeia de tecnologia de informação e equipamentos eletrônicos, ECMA. Esta iniciativa da Microsoft permite que qualquer empresa interessada possa criar sua própria versão do framework .NET.

A Microsoft aprendeu com os erros da Sun na implementação do Java e ouviu as necessidades dos desenvolvedores. Criou um ambiente padrão, com mecanismos simples que pode se integrar a um sistema operacional com as qualidades do Java, mas com a versatilidade e velocidade do C++. A Microsoft criou uma biblioteca única de classes, com uma organização que facilita e simplifica muito o processo de desenvolvimento de software. O .NET possui o que há de mais moderno e performático para executar aplicações com robustez e segurança. Além disso, pode trabalhar virtualmente com qualquer linguagem de programação. Atualmente já passa de 180 linguagens que podem ser usadas em uma mesma aplicação, inclusive o JAVA. Por isso, entre outras vantagens, o .NET com C# tem se tornado tão popular.

Isso agrada aos desenvolvedores. É possível integrar facilmente um código de Java com C#, ou um código de VB.net com C#. Como todas as linguagens usam o mesmo conjunto de funções o que fica fácil (e barato) treinar um programador em C#. Com isso se tem uma quantidade de mão de obra muito mais abundante. Além disso, a Microsoft possui um arsenal de ferramentas de apoio muito superior ao do JAVA atualmente. Pode-se dizer que, atualmente, o JAVA só consegue competir com a plataforma .NET graças à comunidade de software livre como o Projeto Apache e alguns outros.

O custo de se treinar um profissional .NET é muito mais barato que um profissional em JAVA, pois o programador pode ir caminhando entre as linguagens sem traumas. Afinal, as ferramentas são as mesmas. O que muda é apenas a linguagem.

A Microsoft mantém a estratégia de focar no desenvolvedor, aumentando sua produtividade, gerando ferramentas de qualidade de baixo custo e, a partir de agora, formando uma comunidade de software

livre. Liberando partes do .NET como open source, a Microsoft consegue que seu produto seja hoje, sem dúvida, a plataforma de desenvolvimento de maior alcance no mercado web.

Migração do framework para outras plataformas não Microsoft

A popularização da plataforma é um ponto importante. Agradar aos usuários também. Mas e as empresas que têm sistemas em diversos sistemas operacionais. A padronização do C# e o apoio da Microsoft à comunidade de software livre vêm surpreendendo o mercado de TI. Agora é possível a realização de tarefas que simplesmente não eram inimagináveis com o JAVA. Atualmente o Projeto Mono agrega uma plataforma completa para executar aplicativos em diversas plataformas de hardware e diversos sistemas operacionais como Linux, IOS e outros sistemas (jogos e sistemas dedicados).

O modelo de desenvolvimento distribuído, colaborativo e de iniciativa open source

O Projeto Mono nasceu como software livre. Seu criador, Miguel de Icaza, acredita que esse seja o ambiente mais apropriado à inovação. Dessa forma as empresas poderão criar novos produtos e serviços em um modelo cooperativo sem deixar de serem mais competitivas.

A forma de licenciamento permite que as empresas contribuam com melhorias para o código do Mono e ainda assim usem essas melhorias em produtos fechados e licenciando os mesmos da forma tradicional, caso queiram. Este modelo permite uma grande flexibilidade para as empresas que querem investir no projeto e usar partes do código do Mono em seus produtos sem necessariamente, abrir o código de suas aplicações.

São mais de 2500 profissionais envolvidos no desenvolvimento do projeto mono que, conforme citado anteriormente, passa das seis milhões de linhas de código. Alguém poderia questionar a qualidade do código em um projeto deste tamanho, mas é importante entender que essa comunidade não nasceu de um dia para o outro. É uma comunidade que vem se formando através dos últimos 10 anos. Os desenvolvedores recebem os direitos de acesso em função dos seus próprios méritos e pelo elo de confiança que se cria entre os membros. Cerca de 400 desenvolvedores têm direito de alterar o código, mas apenas algo em torno de 80 são contratados em tempo integral para trabalhar no projeto em empresas como a Novell e a MainSoft.

Além disso, o projeto tem processos rígidos de desenvolvimento para garantir a qualidade do Mono. A primeira regra é que nenhum código novo é aceito que não tenha um teste automático vinculado a ele. Esses testes são executados dia e noite nas fazendas de servidores do projeto, gerando relatórios de bugs e enviando informações por e-mail para os responsáveis de cada área no projeto. Os builds podem ser consultados em tempo real por qualquer pessoa na Internet.

O nível de conhecimento das pessoas envolvidas no projeto é muito alto e é possível afirmar que estão praticamente empatados com a equipe da própria Microsoft. Entretanto, a Microsoft conta com uma equipe dedicada de desenvolvedores que é praticamente 10x maior que a equipe do Projeto Mono (quase 4000 profissionais). Acredita-se que este “empate técnico” não se dá apenas pela qualidade dos profissionais do Mono, mas porque todos que doam seu tempo ao projeto estão motivados pelo desafio de criar um projeto de sucesso e também pelo crescimento pessoal que o acesso ao código do Projeto Mono oferece. Para exemplificar, o criador e líder do projeto, Miguel de Icaza, é considerado um dos melhores programadores do mundo. Recebeu o reconhecimento da Microsoft com o título de MVP C# por sua contribuição em código C#. Afinal o Miguel já escreveu sozinho, mais de 1.5 milhão de linhas de código em C#.

Cases de sucesso

Case Daruma

A Daruma é uma grande fabricante de hardware para automação comercial. Seus produtos são um caso claro do uso da interoperabilidade do .NET para expansão da base potencial de clientes. Desde março de 2006 a Daruma passou a suportar o uso de suas impressoras em ambiente Mono com Linux. Foram feitas melhorias internas na estrutura dos drivers Linux das impressoras para ficarem 100% compatíveis com Mono. Com isso qualquer desenvolvedor de aplicações de automação comercial que use o .NET podem usar os equipamentos da Daruma, pois o funcionamento é garantido no ambiente Linux.

Case FluidPlay Games

A FluidPlay Studios é uma empresa anglo-brasileira dedicada a desenvolvimento de jogos 3D para plataforma iPhone e PC para o mercado mundial, com foco em jogos semi-causais, que possam finalmente combinar os gráficos de última geração com a jogabilidade suave e de entretenimento imediato dos jogos clássicos das décadas de 80 e 90. Torment é o atual projeto em desenvolvimento e tem lançamento previsto para abril de 2011 nas plataformas PC e iPhone.

A empresa optou como base para o desenvolvimento a ferramenta Unity3D com linguagem C#. Esta ferramenta é totalmente baseada no Mono, fornecendo ferramentas de porte para PC, iPhone, Wii e brevemente Androide. Segundo Breno Azevedo, desenvolvedor de games com três títulos internacionais com o Battle of Middle Earth e Commande&Conquer III, sócio e Game Director da empresa, *"a simplicidade da ferramenta é tão grande que me permite fazer mudanças no código em tempo real com o jogo em funcionamento, mudando as artes, texturas, animações e comportamentos. Mesmo sendo minha atuação principal na área de design e não em programação*".

A ferramenta trabalha com três linguagens Boo, UnitScript e C#, a empresa optou pelo C# por ser uma linguagem comercialmente aceita e muito simples de usar. Francis Silveira é um dos desenvolvedores da empresa e foi campeão do último Microsoft Students to Bussiness ocorrido no Sergipe ParqueTech. Atualmente ele trabalha com o C# usando um Mac Book, segundo ele "*é muito simples desenvolver para várias plataformas como iPhone e PC tomando cuidados apenas com as limitações impostas pela Apple e modificando apenas os comandos de entrada como teclado, joystick e iPhone touch screen*".

Este é realmente um case de como a interoperabilidade permitida pela plataforma .NET estar permitindo criar empresas inovadoras de alta tecnologia. Criando novos modelos de negócio e usando outras plataformas não Microsoft para agregar valor a produtos.

Case SecondLife

A SencondLife mudou do LSL (Linden Scripting Language) para Mono por ser de 50 a 300x mais rápido. O resultado em ganho de velocidade no SecondLife usando o Mono pode ser visto no youtube em
http://www.youtube.com/watch?v=cGoM9p7q1jk&feature=player_embedded#at=35

O Mono no SecondLife representa um upgrade no simulador que pode acelerar drasticamente a execução de scripts - especialmente aqueles de cálculo intensivo. O código LSL não foi alterado para garantir que todos os objetos existentes no script e anexos continuassem a funcionar como antes, só que agora eles são capazes de rodar mais rápido.

Neste links podem ser vistos vídeos do mono em ação no SencodLife:
http://wiki.secondlife.com/wiki/Mono_demos

Case Mindtouch

A MindTouch é uma empresa que desenvolvem uma das melhores ferramenta de gestão de conhecimento do mundo chamada DekiWiki. Baseada da ideia da Wikipedia, a empresa criou seu próprio sistema de Wiki em C# e como opensource rodando tanto em plataforma Windows como Linux. Esta ferramenta possibilita a construção de conteúdo de forma colaborativa integrado centenas de ferramentas de web 2.0 como twitter, google map, diig, facebook, sistema de BI, geração de relatórios, integração com desktop Windows, gestão e versionamento de documentos.
http://www.mindtouch.com/

Alguns Clientes : AutoDesk, Microsoft, Fujtsu, The Washinton Post, HTC, Palm, Novell, VIACOM, EMC, PayPAL, MTV, Toyota, Governo da Australia, Harvard University , U.S. Army , Nasa , Cisco , The New York Times, Panasonic, AVAYA, HP.

Case Cidade de Munich

A maior migração já feita para Linux foi feita com Mono no Software ActiveEntry - Software para provisão e gerenciamento usando pelo governo Alemão da cidade de Munich. Era uma aplicação VB6 com certa de 7 milhões de linhas de código. Essa aplicação foi reescrita com Mono e Linguagem C# ficando com apenas 2 milhões de linhas de código. Todo os servidores Windows foram migrados para Suse Linux. http://www.novell.com/success/volcker.html http://www.activeentry.de/

"*Mono is now a critical part of our cross-platform development*," said Matthias Bauer, head of development for Völcker Informatik AG. *"Using Mono was the only way for us to leverage our existing software and give our customers what they need on Linux*."

Atualmente a aplicação consome o uso de 350 Servidores, são 40mil estações de trabalho que atendem a cerca de 150mil usuários.

Case Pollares

A empresa Pollares é uma empresa especializada no desenvolvimento de sistemas web e tem um case interessante de migração do ambiente de desenvolvimento para o MacOS X. A empresa adota como plataforma para os desktop notebook Mac . Um dos problemas enfrentados era executar o Visual Studio 2008 em um windows XP virtualizado. Além da demora na carga dos aplicativos para desenvolvimento, funcionalidades como intelisense e publicação das aplicações no Windows Serve eram prejudicadas. Em alguns momentos pelo excesso de processamento ocasionava queda da máquina virtual.

A empresa optou por migrar para o ambiente Mono com MonoDevelop, este foi capaz de abrir as soluções do Visual Studio sem problemas. Os problemas de performance e velocidade da IDE foram superados, e alguns ganhos como a integração eficiente com o servidor de código Subversion adotado pela empresa facilitou bastante o trabalho. Segundo o diretor de tecnologia da empresa Sr. Carlos Eduardo Lopes , "todas as bibliotecas do projeto foram importadas não gerando problema algum e nós utilizamos o xsp2 server integrado no MonoDevelop para os testes da aplicação e tudo funciona perfeitamente. ".

Em relação à migração foi necessário migrar o formato dos arquivos para o formato do Mac e foram feitos pequenos ajustes na aplicação em alguns controles AJAX.NET. Devido a um problema na atual versão estável do Mono, mas que já está resolvido na versão em desenvolvimento. Mas seguindo as orientações disponibilizadas no site do projeto foi possível resolver os problemas.

Hoje a aplicação é compilada pelo Mono e publicada em um windows Server com SQL Server, segundo o Sr. Carlos, "Aqui na empresa nós utilizamos o Windows Server 2008 para servidor de aplicações, e isso não foi problema algum após a nossa migração, tudo funciona perfeitamente no IIS7."

Nível de compatibilidade com o Framework .Net

O Mono possui compatibilidade binária com o .NET. Isso significa que não é necessário recompilar o código para executar em outras plataformas não Windows. Basta copiar seus executáveis ou bibliotecas (DLL) para o sistema desejado e usar o Mono para executá-los. Por exemplo, digamos que seu aplicativo seja chamado de Teste.exe basta invocar o programa Mono e passar como parâmetro o nome do seu aplicativo ex: "Mono Texte.exe". Em algumas versões de Linux, por exemplo, como no caso do Suse Linux o Mono já vem previamente instalado e no momento que você realiza um duplo clique em arquivo executável .NET o sistema operacional automaticamente já invoca o Mono e executa seu aplicativo de maneira fácil e descomplicada.

Com o Mono é possível executar aplicações .NET em diversos sistemas operacionais e plataformas de hardware não suportadas oficialmente pela Microsoft. A figura abaixo ilustra todas as arquiteturas suportadas pelo Mono.

Supported Architectures	Runtime	Operating system
s390, s390x (32 and 64 bits)	JIT	Linux
SPARC (32)	JIT	Solaris, Linux
PowerPC	JIT	Linux, Mac OSX, Wii, PlayStation 3
x86	JIT	Linux, FreeBSD, OpenBSD, NetBSD, Microsoft Windows, Solaris, OS X
x86-64: AMD64 and EM64T (64 bit)	JIT	Linux, Solaris
IA64 Itanium2 (64 bit)	JIT	Linux
ARM: little and big endian	JIT	Linux (both old and new ABI), iPhone
Alpha	JIT	Linux
MIPS	JIT	Linux
HPPA	JIT	Linux

Plataformas suportadas pelo Mono

Oficialmente o Mono suporta até a plataforma .NET 4.0, com exceção de algumas bibliotecas como o WPF, EntityFrameWork , WF e algumas limitações para WCF.

Podemos ressaltar:

- Suporte ao LLVM para melhorar o desempenho de cargas computacionais e de servidor.
- Suporte a Java, com integração do código GPL do JAVA o que garante 100% de compatibilidade com código open source de Java .
- Ruby e Python sendo esta uma contribuição de código da Microsoft.
- LINQ para SQL usando o projeto DBLINQ que é uma implementação open source do Linq para bancos de dados livres como MySql, PostGreSQL e FireBird que foi integrado ao Mono.
- Novo debugger integrado com o MonoDevelop e com o MacOS X.
- O CSHARP Shell agora suporta auto complete.
- Esta versão está integrando as iniciativas de código aberto da Microsoft com a disponibilização dos códigos do Ajax.NET, ASP.NET MVC1, MVC2, Silverligth ToolKit, Silverligth Tests Managed Extensibility Framework (MEF) , System.Data.Services.Client (OData), Dynamic Language Runtime.

Migração do WebISS para Mainframe

Pela análise feita no sistema WebISS não existe nenhuma questão impeditiva à migração da plataforma Windows para Linux rodando em mainframe. Toda a tecnologia desenvolvida para o sistema WebISS está disponível e estável no Mono. Com exceção de alguns componentes de terceiros que fazem uso de algumas funcionalidades do Windows para funcionar e que terão que ser substituídas. Vamos observar cada módulo:

Aplicação Web – Interface de usuário

Este é o núcleo do sistema e está escrito em linguagem C# 2.0 e Asp.Net. Ambas as tecnologias estão totalmente mapeadas pelo Mono. Entretanto, a toda a emissão de relatórios e saídas formatadas para impressora e geração de arquivos pdf do WebISS utilizam um componente de terceiros que se vale da API Win32 (o Component One). Este componente deverá ser reescrito em C# 2.0 ou substituído por outra ferramenta que não acesse a API Win32.

Para atualização dinâmica das páginas é utilizado o componente MagicAjax. Este componente precisa ser testado no ambiente do Mono ou substituído pelo Ajax.NET da Microsoft que é open source e já está totalmente integrado ao Mono.

A terceira peça do sistema que precisa de atenção é o acesso ao banco de dados à camada de persistência que salva no banco de dados SQL Server. Como é usada uma camada de persistência própria para acessar os dados no SQL Server, basta adaptá-la para acessar os dados no DB2.

Não é uma tarefa complexa, mas é uma tarefa que exigirá bastante trabalho. Sugere-se que sejam criados sistemas de testes automatizados para testar todo o processo de acesso ao banco de dados. Com isso garantiremos que o sistema estará funcionando adequadamente nos dois ambientes.

DeS WebService

É a parte de sistema que provê a estrutura de comunicação com os aplicativos DeS pela Internet. Esta aplicação está escrita em C# 2.0 e funcionando sob a plataforma do IIS Server da Microsoft. A parte de webserver está a anos muito estável no Mono. O desafio aqui é colocar em funcionamento a aplicação com o servidor Apache do Linux usando o ModMono ou o FastCGI.

Não é uma atividade complexa, mas precisam ser feitos testes de carga e estabilidade do serviço para que se possa garantir o perfeito funcionamento do sistema.

WSNFS-e

Sistema de distribuição de notas eletrônicas escrito em C# 3.5 com WCF. Apesar do C# 3.5 estar suportado, o WFC no Mono ainda não está completo. É possível testar o serviço, mas não existe garantia que o sistema vai funcionar corretamente. A primeira opção é testar a implementação e verificar se ela funciona corretamente. Analisar os problemas que podem estar ocorrendo e verificar a possibilidade que corrigi-los e adaptar o código diretamente no mono. A opção é reescrever estes serviços.

G6TaskService

Este é um serviço de processamento de dados que roda em *batches*, escrito em C# 2.0 e faz uso de dados acessados diretamente do SQL Server. Este serviço pode rodar facilmente no Linux sem muitas diferenças do ambiente Windows.

Dificuldades, obstáculos e ajustes de compatibilidade

Funcionamento adequado do Mono no MainFrame

Atualmente as versões do framework .NET usadas no sistema WebISS #2.0 e #3.5 estão estáveis no Mono. Entretanto, a literatura não garante que o ambiente do mainframe oferece o desempenho desejado. O código atual está otimizado para mainframes da família z9. Portanto, são necessários esforços para habilitar o JIT para o z10. É necessário executar testes de carga para verificar os pontos que merecem atenção ou que oferecem oportunidades de melhoria no ambiente do mainframe.

Disponibilidade de profissionais qualificados em Mono e Mainframe

Problemas relativos à máquina virtual do mono no mainframe são questões extremamente específicas e um risco do projeto é não conseguir alocar profissionais que tenham esta competência. Felizmente, temos alguns contatos no Brasil e no exterior que podem ajudar.

Atualmente não existe suporte comercial disponível na Novell, tampouco pela IBM. Acreditamos que o case gerado pela WebFis vai gerar mais visibilidade para o projeto Mono em mainframe e que possa ser criada uma parceria com a Novell para desenvolvimento em conjunto. Ao conversarmos com o líder do Projeto Mono, Miguel de Icaza, ficou claro o compromisso em ajudar na solução de problemas usando os canais da comunidade no projeto e até mesmo disponibilizar profissionais no caso de *bugs* que apareçam durante o porte da aplicação.

Alternativa ao Projeto Mono: Migração completa para Java

Uma possibilidade é a migração do sistema par a plataforma java. Existem três caminhos neste sentido.

- Reescrever completamente a aplicação em Linguagem Java
- Conversão de parte da aplicação .NET para JAVA com IKVM
- Execução da aplicação .NET em ambiente JAVA

Reescrever completamente a aplicação em Linguagem Java

Reescrever a aplicação em JAVA implica que um reinvestimento completo. O perfil da equipe e as tecnologias são completamente distintas. Apesar da plataforma JAVA estar consolidada no mercado, foram criadas muitas vertentes de desenvolvimento open source. O resultado disso foi que a comunidade se dividiu em várias doutrinas e criando centenas de bibliotecas.

Esse fracionamento vem tornando os projetos cada vez mais complexos e tem exigido mão de obra cada vez mais qualificada. O impacto deste fracionamento da tecnologia aparece nos prazos e custos dos sistemas. Nada de código poderia ser aproveitado, sendo necessário reescrever todo o software já pronto.

Conversão de parte da aplicação .NET para JAVA com IKVM

Uma opção é converter parte da aplicação .NET diretamente para JAVA. Ainda assim, toda a interface web com o usuário terá que ser refeita.

Execução da aplicação .NET em ambiente JAVA

A terceira opção viável é a execução de toda a ASP.NET no ambiente de execução JAVA usando o servidor TOMCAT, JBoss ou mesmo o WebSphere da IBM usando o Grasshopper da MainSoft (http://dev.mainsoft.com/). O desenvolvimento e licenças de tempo de execução já estão disponíveis e livres de encargos, sem restrições em relação à quantidade de CPU's ou uso comercial.

A grande vantagem deste modelo é não precisar reescrever nenhum código, aproveitando toda base de código já escrito e executar a aplicação no ambiente JAVA já consolidado em ambiente mainframe. Mas ainda sim serão necessários remover ou reescrever os componentes que não são 100% .NET e também a camada de acesso a banco de dados (no caso, para o DB2).

Quadro de Atividades para migração para ambiente Java :

Atividade	Migração Mono	Migração JAVA	Execução em JAVA com Grasshopper
Formação e Treinamento de Equipe em JAVA		X	
Construção da Arquitetura do aplicativo em JAVA		X	
Reconstrução das todas telas do sistema		X	
Reconstrução de toda Lógica de Negócio		X	
Migração da Base de Dados	X	X	X
Migração da camada de persistência de Dados	X	X	X
Adaptação dos componentes de geração de relatórios	X	X	X
Criação de teste automatizados da camada do Banco de Dados para garantir compatibilidade da migração	X	X	X
Teste das telas	X	X	X
Teste do modelo de negócio		X	