5
Ferramentas de Apoio

Os softwares desenvolvidos atualmente estão cada vez mais complexos, com regras de negócios mais elaboradas, feitos para serem executados em diferentes arquiteturas e sob o requisito de estarem sempre disponíveis na Internet. Acompanhando esta evolução do software, surgiram nos últimos anos várias ferramentas de auxílio ao desenvolvimento.

Essas ferramentas são utilizadas para automatizar o processo de testes (unitários, de integração, de carga, de performance, entre outros), medir a cobertura do código atingida pelos testes, controlar a versão do software em desenvolvimento, registrar e acompanhar pendências (*issue trackers*) ou mesmo o *Backlog*, entre outras.

Quando usadas da forma correta e por pessoas que sabem manipulá-las, elas podem ser muito úteis ao processo de desenvolvimento, pois aceleram o trabalho a ser realizado. Portanto, um dos objetivos deste estudo é a escolha de um conjunto de ferramentas que possa auxiliar o trabalho desenvolvido na organização estudada. A estrutura das ferramentas utilizadas na organização estudada era precária (não havia nem mesmo um controle de versão do código produzido) e as ferramentas não eram integradas.

Este capítulo, então, descreve um primeiro conjunto de ferramentas que poderiam ser utilizadas para melhorar a infraestrutura da unidade estudada. Porém, esta não é uma lista exaustiva e fechada, uma vez que algumas ferramentas novas ferramentas foram adicionadas nos capítulos seguintes, conforme se descobria que seu uso poderia ser interessante. Há também algumas ferramentas citadas neste capítulo que foram utilizadas ao longo do trabalho, porém, o seu uso não é de grande relevância, portanto, não serão mais citadas nos próximos capítulos.

5.1.Introdução

Sabe-se que ferramentas por si só não garantem o sucesso de um projeto e algumas vezes podem até ser prejudiciais ao trabalho – se suas funcionalidades não atenderem a seus usuários ou se a sua complexidade atrasar ou desestimular o trabalho da equipe. Porém, quando bem escolhidas e após o treinamento necessário, as ferramentas podem auxiliar o trabalho diário de desenvolvedores.

Deste modo, se os usuários têm o talento mínimo esperado pela ferramenta, ela nem prejudica nem ajuda. Se os usuários têm talento abaixo do mínimo, a ferramenta atrapalha. Se tiverem acima do mínimo, a ferramenta ajuda tanto mais quanto mais acima do mínimo estiver o usuário. Na realidade isso implica a necessidade de formação, e treinamento no uso da ferramenta. Talvez experiência profissional também seja importante para o bom uso da ferramenta.

Quando um conjunto de ferramentas passa a ser utilizado por uma equipe, sabe-se que há um período de adaptações e mudanças que pode diminuir a produtividade temporariamente – a chamada curva de aprendizado. Porém, passado este período, é possível se notar os benefícios da utilização de ferramentas adequadas.

Este trabalho procurou estudar e implantar uma série de ferramentas que auxiliam o desenvolvimento de aplicações em ColdFusion (http://www.adobe.com/products/coldfusion/).

5.2.ColdFusion

ColdFusion é uma plataforma para rápido desenvolvimento de sistemas *Web* e baseada em Java, criada em 1995 por Jeremy Allaire e Joseph Allaire. Foi inicialmente desenvolvida para facilitar a integração entre páginas HTML e o banco de dados, porém transformou-se em um ambiente completo de desenvolvimento, contendo até mesmo uma linguagem de *script*.

A linguagem de *script* associada ao ColdFusion, ColdFusion Markup Language (CFML) assemelha-se a HTML, uma vez que é formada por *tags*; e é funcionalmente comparada a linguagens como ASP, PHP e JSP. ColdFusion também suporta outras linguagens de programação e linguagens de script como JavaScript e CFScript.

A principal vantagem de utilizar ColdFusion é a pequena curva de aprendizado para utilização desta ferramenta. Apesar de ser uma ferramenta proprietária, a alta produtividade atingida com sua ajuda justifica o investimento inicial.

ColdFusion possui uma série de componentes e tags que podem ser utilizados com facilidade para realizar tarefas que precisariam de muitas linhas de código. Por exemplo, no ColdFusion 9, há uma *tag* que torna a utilização de Google Maps muito simples: <cfmap>. Portanto, o desenvolvedor não precisa aprender a usar a API do Google e escrever um extenso código para obter um mapa; ele pode usar a *tag*, passando os parâmetros necessários.

5.3.Ambiente Integrado de Desenvolvimento (IDE)

O ambiente integrado de desenvolvimento (IDE) é a ferramenta mais importante que um desenvolvedor utiliza, uma vez que tenta reunir em um só lugar várias funcionalidades auxiliares ao desenvolvimento de software: editores para escrita do programa, execução de testes, integração com sistemas de versionamento, ferramentas de debug, etc. Os ambientes de desenvolvimento utilizados na organização são o Dreamweaver e o Eclipse.

5.3.1.Dreamweaver 3

Dreamweaver (http://www.adobe.com/) é uma aplicação para desenvolvimento web criada pela Macromedia, que desde 2005 pertence à Adobe. Esta ferramenta requer compra de licença para uso, está disponível em vários idiomas e suporta o desenvolvimento em várias linguagens, entre elas: PHP, ASP, CSS, HTML, JSP, VB, XHTML, Coldfusion, Java, etc. Ela oferece a possibilidade de edição gráfica, além da textual. Portanto, pessoas sem conhecimento de código podem utilizá-la para desenvolver *sites*.

Esta é a ferramenta oficial utilizada pela instituição estudada e vem sendo usada há bastante tempo. Porém, é uma ferramenta muito pesada e muito antiga – as primeiras versões datam de 1997 – que não atende às atuais necessidades dos desenvolvedores.

Um dos pontos mais importantes na estruturação do ambiente de desenvolvimento feito por este trabalho é a utilização de um controle de versão,

que não estava sendo feito. Para esta tarefa, foi escolhido o uso do Subversion (http://subversion.tigris.org/) e se tornou necessário verificar como o Dreamweaver 3 (CS3) e o Subversion poderiam se integrar, uma vez que é extremamente interessante a utilização de um controle de versão dentro do próprio ambiente de desenvolvimento.

Verificou-se que há alguns *plugins* que prometem fazer esta integração: SVN4DW e SubWeaver, por exemplo. Infelizmente, ambos os testes realizados com estes *plugins* não obtiveram sucesso. Eles não funcionavam como o esperado e também é muito comum encontrar reclamações em fóruns relacionadas a estes *plugins*.

5.3.2.Dreamweaver 4

O Dreamweaver 4 (CS4) é a nova versão desta ferramenta, lançada pela Adobe. Entre o conjunto de funcionalidades adicionadas, tem-se a possibilidade de integração com o Subversion como principal mudança. Porém, esta integração deixa muito a desejar, uma vez que nem todas as funcionalidades do Subversion são aqui implementadas – por exemplo, o *diff*, funcionalidade que compara duas versões de arquivo, ressaltando os pontos de alteração. Portanto, um *upgrade* para esta versão não é aconselhado, uma vez que o principal motivo de migração seria a possibilidade de utilização de controle de versão – que por sua vez não funciona da maneira como a esperada. Por outro lado, as atividades básicas de versionamento, como a realização de *commit* e *update* estão presentes no CS4. Ou seja, com a utilização desta ferramenta, seria viável realizar um controle de versão. Além do mais, os editores – que também fazem parte da equipe de Internet e que já estavam acostumados ao CS3 – não sentiriam muito impacto ao migrar para o CS4.

5.3.3.Eclipse

Eclipse (http://www.eclipse.org/) é um ambiente para desenvolvimento de software utilizado principalmente por desenvolvedores Java. Porém, esta ferramenta pode ser utilizada para o desenvolvimento em muitas linguagens distintas e serve de base para muitos outros ambientes de desenvolvimento, como Flex Builder (http://www.adobe.com/products/flex/), Aptana

RadRails (http://www.radrails.org/) e Coldfusion Builder (http://www.adobe.com/products/coldfusion/cfbuilder/ features/).

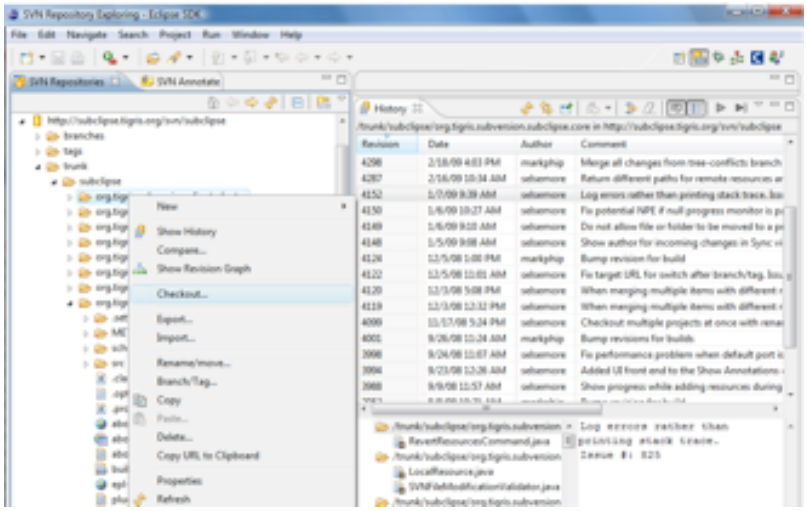

Figura 7 – Ilustração do SVN integrado ao Eclipse (*checkout* de projeto).

O Eclipse, por ser uma ferramenta largamente utilizada e *open source*, possui uma infinidade de *plugins*, dos quais três são utilizados neste trabalho para compor uma alternativa *open source* de ambiente de desenvolvimento: Aptana Studio (http://www.aptana.org/), CFEclipse (http://www. cfeclipse.org/) e Subclipse (http://subclipse.tigris.org/).

CFEclipse é um *plugin* para o desenvolvimento de programas em Coldfusion no Eclipse. Este *plugin* fornece a funcionalidade de *auto-complete* da sintaxe do Coldfusion, além de fazer validações de código.

Aptana Studio é um *plugin* para Eclipse que facilita o desenvolvimento *web*, pois dá suporte às sintaxes de HTML, CSS e Javascript. Também neste *plugin* há a funcionalidade de transferência de arquivos via FTP, o que auxilia a gerência dos inúmeros servidores que a organização possui.

Subclipse é um *plugin open source* para integração ao Subversion (sistema de controle de versão) pelo Eclipse. Desta maneira, dentro do próprio ambiente de desenvolvimento, é possível manter o histórico de alterações do projeto, indicando as modificações realizadas em cada nova versão.

O Subclipse pode ser instalado por meio do menu de instalação de novos *plugins* do Eclipse (em Help -> Software Updates -> Available Software -> Add Site... -> Instal...).

Para começar a utilizar o Subclipse, é necessário criar um repositório de dados e adicionar os arquivos desejados. Em seguida, faz-se um *checkout* do projeto, e se pode iniciar a fazer alterações nos arquivos do projeto. O Subversion irá controlar as modificações realizadas.

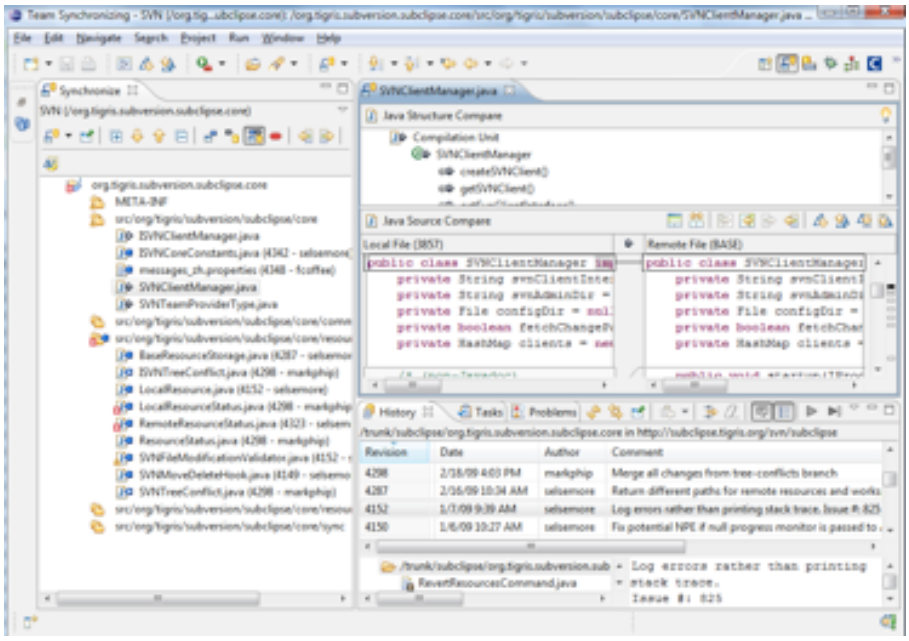

Figura 8 – Ilustração das mensagens que aparecem no *console* e das alterações sendo marcadas à esquerda.

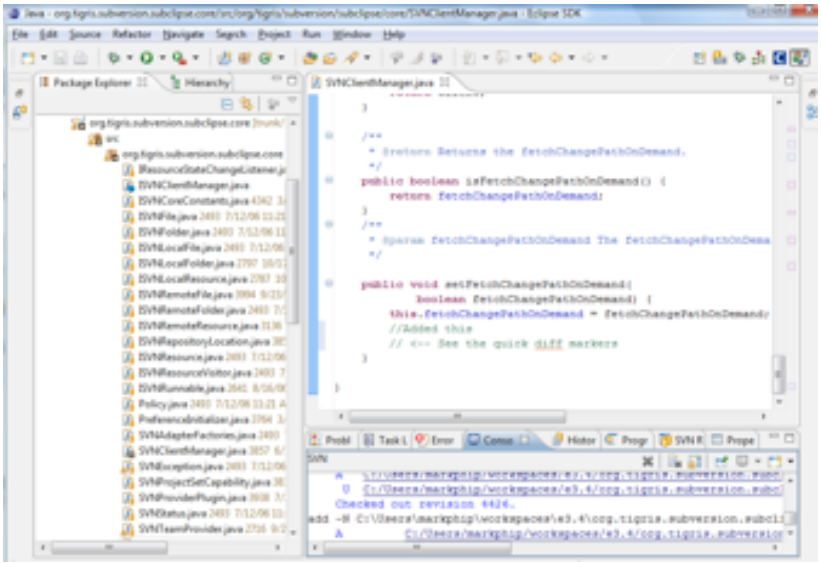

Figura 9 – Ilustração de um *update* feito pelo Subclipse.

Ao término de cada implementação, pode-se fazer um *checkin* das modificações realizadas, para ser gerada uma nova versão do software em

desenvolvimento. Ao se fazer um *checkin*, deve-se escrever um comentário, explicando o motivo da alteração realizada. Da mesma forma, para buscar as alterações realizadas, deve-se fazer um *update* do projeto e o SVN mostrará os arquivos que serão atualizados. Estes passos podem ser vistos nas Figuras 7, 8, 9 e 10.

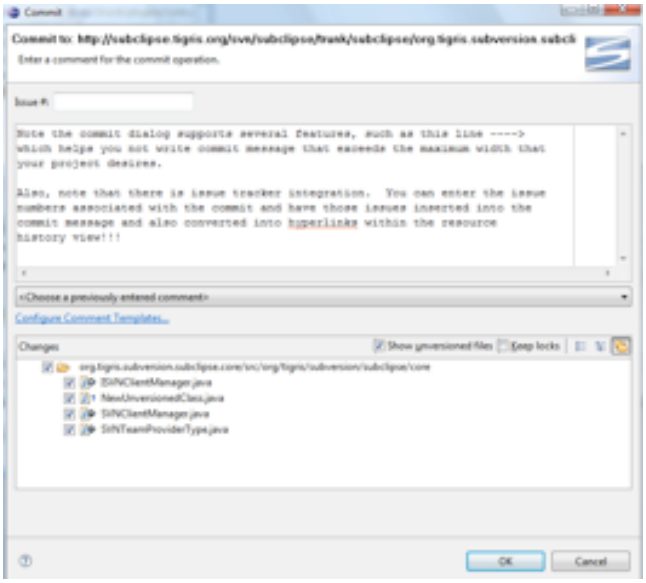

Figura 10 – Ilustração do dialogo de *commit*.

5.3.4. ColdFusion Builder

Coldfusion Builder é um ambiente integrado de desenvolvimento criado pela Adobe e baseado no Eclipse, para o desenvolvimento em Coldfusion.

Esta ferramenta é muito interessante para o desenvolvimento em Coldfusion. Ela dá suporte a várias sintaxes *web*, como HTML, CSS e Javascript, com recursos de *auto-complete*. Também possui um sistema de transferência via FTP que facilita bastante a atualização dos *websites* em diferentes servidores. Por último, ela possui a funcionalidade de *tracking* de funções, isto é, a partir da chamada de uma função, ele oferece a possibilidade de ida ao componente onde a função está declarada. E também, uma vez que é baseado no Eclipse, os *plugins* desenvolvidos para este podem ser integrados ao Coldfusion Builder.

Sabendo-se que a equipe de Internet da organização é bastante heterogênea (há desenvolvedores, designers e editores) a escolha da ferramenta a ser utilizada ficou a cargo da própria pessoa. Como o Dreamweaver 4 e Coldfusion Builder tem a possibilidade de utilização de Subversion, a escolha entre estas duas

ferramentas deve ser baseada na experiência de cada integrante da equipe, a fim de minimizar o impacto causado pela curva de aprendizado de uma nova ferramenta.

5.4.*Plugins* para Firefox

5.4.1.Filezilla Client

FileZilla Client (http://filezilla-project.org/) é uma ferramenta *open source* de transferência de arquivos via FTP, FTPS e SFTP. É muito fácil de ser instalada e usada. Possui suporte a vários protocolos e funciona em diferentes sistemas operacionais, como Windows, Mac e Linux. Está disponível em várias linguagens e suporta a transferência de arquivos pequenos e grandes (até 4GB). Esta ferramenta é muito usada para a atualização do *website* da organização, pois os arquivos são enviados aos diversos servidores via FTP.

5.4.2.FireBug

FireBug (FireBug) é um *plugin open source* para Firefox muito útil na criação de páginas utilizando CSS, HTML e Javascript. Com ele é possível alterar, "debugar" e monitorar em qualquer página no próprio Firefox o conteúdo relacionado a estas tecnologias. Com o FireBug pode-se também monitorar a atividade de rede, como por exemplo, quanto tempo demora para carregar uma determinada página, verificar as requisições feitas ao servidor e suas respostas.

O suporte a CSS do FireBug é muito interessante, com ele é possível visualizar todas as informações de CSS referentes a todos os componentes da página e até mesmo alterá-las, vendo o resultado no próprio browser. Também é possível visualizar todas as informações do DOM (*Document Object Model*), o que torna mais fácil a utilização de tecnologias como Ajax e jQuery – ambas muito utilizadas no *website* da organização.

A alteração de CSS no *website* não é muito comum, uma vez que este possui uma identidade visual que não pode ser muito alterada. Porém, em algumas situações especiais, o CSS é alterado e nestes momentos o uso do FireBug se torna indispensável.

5.4.3.HTML Validator

Há um outro *plugin* para FireFox, o HTML Validator (HTML Validator), que valida as páginas no próprio *browser*. Com ele é possível verificar se há erros ou pequenos problemas de HTML na página, quantos são, que tipo de erro e qual a possível correção para os mesmos. O algoritmo utilizado para verificação é o mesmo do presente na página validator.w3.org, pertencente à W3C – principal organização internacional da *World Wide Web* (WWW).

5.5.Ferramentas de Apoio a Testes

Há um outro *plugin* para FireFox, o HTML Validator (HTML Validator), que valida as páginas no próprio *browser*. Com ele é possível verificar se há erros ou pequenos problemas de HTML na página, quantos são, que tipo de erro e qual a possível correção para os mesmos. O algoritmo utilizado para verificação é o mesmo do presente na página validator.w3.org, pertencente à W3C – principal organização internacional da *World Wide Web* (WWW).

5.5.1.CFUnit

O CFUnit (http://cfunit.sourceforge.net/) é um *framework* para teste unitário baseado no JUnit, porém desenvolvido para aplicações em Coldfusion. Com ele é possível testar as funções criadas nos componentes do Coldfusion (arquivos com extensão cfc), do mesmo modo em que se pode testar métodos de classes Java.

Para configurar o *framework* é bastante simples, basta baixar o código e deixá-lo no mesmo diretório do resto da aplicação. Porém, é interessante integrar estes testes unitários com ferramentas de build automático, para que a execução dos testes não seja uma tarefa manual.

Utilizando-se o Eclipse, o *plugin* CFEclipse e a extensão CFUnit-Ant (http://cfunit.sourceforge.net/help-ant.php), é possível configurar o projeto para que os testes unitários sejam executados por dentro da própria IDE, de maneira mais eficiente.

O CFUnit implementa as principais funções encontradas nos *frameworks* xUnit; nele é possível encontrar: assertTrue(), assertEquals(), assertFalse(), assertSame(), assertOutputs(), fail().

5.5.2.JMeter

O JMeter (http://jakarta.apache.org/jmeter/) é uma ferramenta *open source* utilizada para testes de performance e de carga, pertencente ao Projeto Apache Jakarta. Ela pode ser utilizada para simular um grande número de acessos simultâneos ao aplicativo a ser testado, gerando, ao final da execução, diferentes tipos de relatórios (inclusive gráficos). Por ser escrito em Java, é altamente portável.

Figura 11 – Parâmetros de configuração do *Thread Group*.

Ele pode ser usado para teste de carga e performance em diferentes tipos de serviços, como Mail (POP3 e IMAP), JMS, LDAP, JDBC, SOAP e WEB (via HTTP e HTTPS).

O JMeter pode ser obtido no site do Apache, em um arquivo zip. Para iniciar o JMeter, deve-se executar o arquivo jmeter.bat, que se encontra dentro da pasta bin do pacote baixado.

A primeira tarefa na criação do plano de testes do JMeter é a definição dos parâmetros do "Thread Group", como indicado na Figura 11. Nesta janela, é possível definir quantas *threads* serão criadas para cada teste executado (simulando múltiplos acessos) e em quanto tempo todas as *threads* serão carregadas, bem como quantas vezes a bateria de testes será executada.

O próximo passo é a construção de uma requisição, que no caso da Figura 12, é do tipo "HTTP Request HTTP Client". Há vários tipos de requisições no JMeter, que devem ser utilizadas de acordo com a necessidade. No exemplo da Figura 12, está sendo testada a criação de um novo objeto.

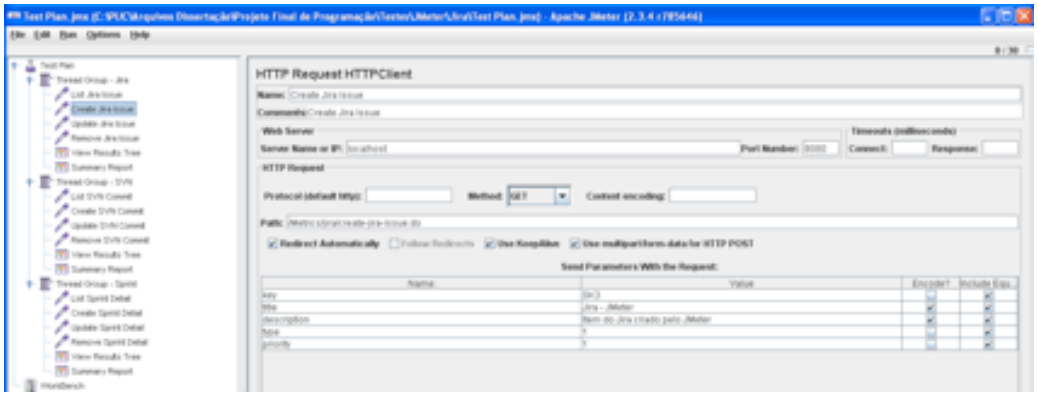

Figura 12 – Figura que representa uma requisição de teste, com o caminho e demais parâmetros configurados.

Nesta janela, deve-se definir o caminho da ação a ser executada (representada pelo parâmetro "Path") e os parâmetros necessários para a execução do teste, representado pela tabela exibida na figura. Estes parâmetros são os mesmos dos atributos contidos na *url* chamada pelo usuário quando o *site* é utilizado.

Após esta parametrização, é possível executar o teste utilizando o comando (Run -> Start). Porém, antes, é importante escolher os tipos de relatórios de saída que serão gerados após a execução dos testes. Nas Figuras 13 e 14, há dois exemplos de relatórios utilizados: resultados em árvore e relatório resumido.

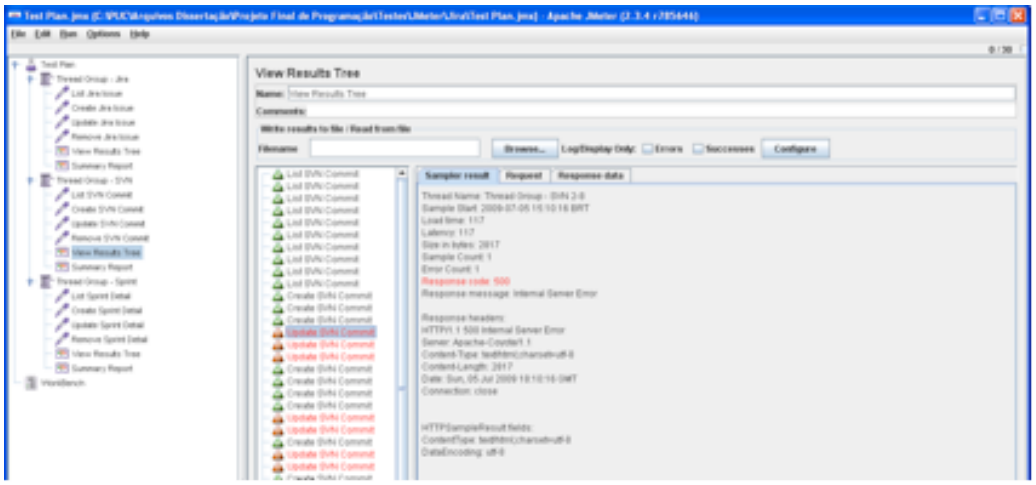

Figura 13 – Relatório em árvore gerado pelo JMeter após a execução de testes.

O resultado em árvore, como pode ser visto na Figura 13, apresenta cada teste executado, colorindo-o de verde caso tenha sido bem-sucedida a execução, e de vermelho, caso contrário. Ao escolhermos um item da lista de testes

executados, maiores informações sobre o mesmo são exibidas, auxiliando a busca por possíveis falhas.

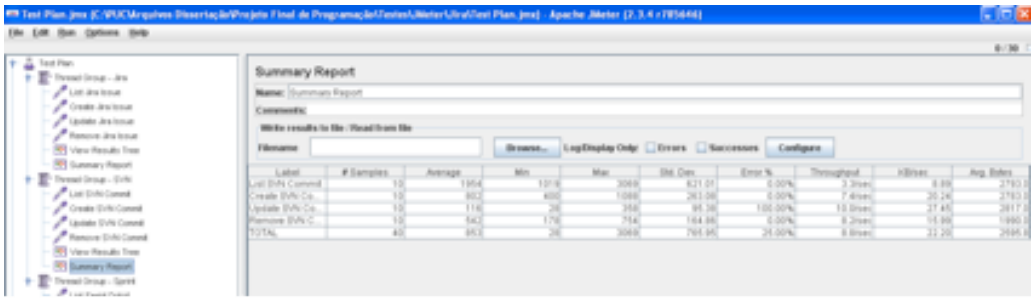

Figura 14 – Figura que ilustra o relatório resumido gerado pelo JMeter após a execução de alguns testes.

Um outro tipo de relatório gerado pelo JMeter, que também é bastante interessante, é o chamado "Relatório Resumido". Ele apresenta em um formato de tabela os resultados atingidos pelo teste. Nesta tabela, pode-se verificar estatísticas de sucesso, falha, bem como de tempo de resposta e taxas de transferência de dados. Este tipo de relatório está representado na Figura 14.

Há, também, as "Assertions" do JMeter, que são pontos de afirmação utilizados para verificar se uma determinada resposta está de acordo com alguma afirmação colocada na requisição.

Nesta ferramenta, a construção dos testes pode ser manual, como explicada nas linhas acima, e também automática, utilizando-se o recurso de "filmagem". Ou seja, o usuário pode executar vários passos no *browser*, que podem ser repetidos pelo JMeter como uma bateria de testes.

O JMeter é uma ferramenta muito útil para este estudo, uma vez que testes de carga e de estresse são fundamentais para o monitoramento do *website* da organização estudada.

5.5.3.Selenium

O Selenium (http://seleniumhq.org/) é uma ferramenta *open source* para automação de testes de aplicativos *web*. Esta ferramenta é altamente integrada ao Firefox, uma vez que existe um *plugin* do Selenium para o Firefox.

Com este *plugin*, é possível acessar um determinado site e gravar os passos executados, para que estes sejam executados posteriormente de uma maneira automática. É possível cadastrar casos de teste e até mesmo suítes de testes, para

serem executadas em conjunto. Quando a execução não gera falhas, é exibida uma linha verde no nome do teste. Caso contrário, é exibida uma linha vermelha, indicando que houve falha.

O Selenium pode ser utilizado em conjunto com o Firefox (o que facilita bastante o seu uso). Assim, deve-se instalar o respectivo *plugin* do Firefox. Feito isto, irá aparecer no menu "Tools"do Firefox, a opção "Selenium IDE".

Assim, para criar um novo caso de teste, deve-se utilizar o botão vermelho de gravação do Selenium e acessar a página a ser testada, executando os passos desejados. O Selenium cria um *script* de execução de testes, que pode ser facilmente alterado por um programa de edição de textos, facilitando a criação de novos testes ou a alteração de determinados parâmetros.

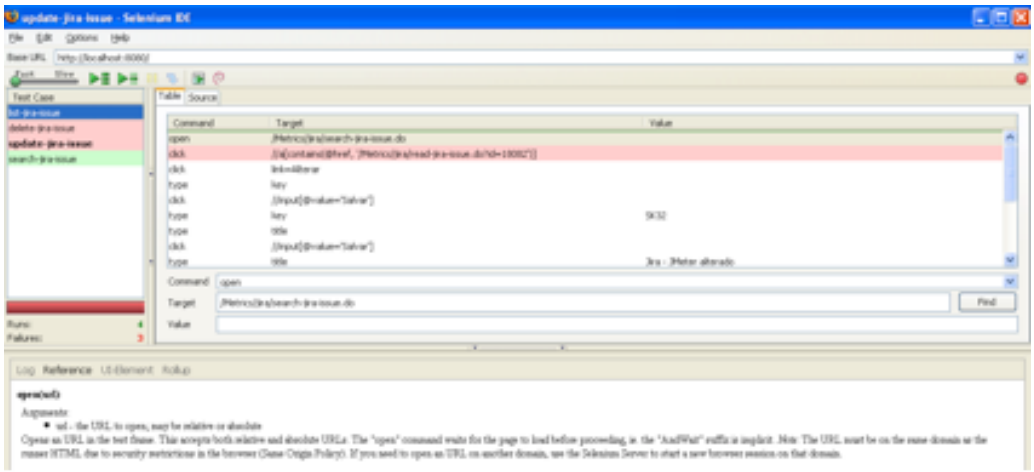

Figura 15 – Utilização do Selenium para testar aplicativos *Web*.

Ele pode gerar os scripts de execução em diversos formatos, até mesmo em Java, para que sejam incorporados a outros tipos de teste, dentro do próprio ambiente de desenvolvimento.

Na Figura 15, tem-se uma ilustração da utilização do Selenium. É possível identificar as execuções que falharam e as que foram bem-sucedidas, também se pode alterar o valor dos parâmetros e criar novos testes, utilizando o botão vermelho do canto superior direito da figura.

5.6.Ferramentas Google

Uma vez que este trabalho está relacionado ao *website* de uma organização, como sendo o principal produto de comunicação externa da mesma, é muito

importante que seja feito um trabalho de promoção do *website*. Este trabalho é um dos principais projetos do ano de 2010 e algumas ferramentas da Google serão indispensáveis nesta fase.

5.6.1.Google Wiki

Como parte da estruturação do departamento, surgiu a necessidade de uma ferramenta *wiki* para armazenar o conhecimento adquirido pelos integrantes da equipe. Assim, cada membro da equipe deve ser responsável pela atualização de seu conteúdo, com o objetivo de melhorar a gestão de conhecimento do departamento.

A ferramenta escolhida foi a Google Wiki, uma vez que é de fácil instalação e uso. Por meio do Google Sites, pode-se criar uma *wiki* em um pequeno espaço de tempo. A atualização de seu conteúdo também é muito simples e, o fato de ser uma ferramenta da plataforma Google também pesou na escolha – uma vez que outras ferramentas da mesma plataforma já são utilizadas pela equipe.

5.6.2.Google Analytics

Google Analytics é uma ferramenta que proporciona muitas informações relativas à popularidade de um *website*. Com ela, é possível identificar o perfil do usuário que acessa o *site* (como por exemplo, seu país e idioma), qual o tempo médio de navegação pelo *site*, entre outras métricas. Esta ferramenta é muito útil para estudar o tráfego do *site*, com objetivo de melhorar a popularidade do mesmo. Está muito relacionada a atividades de marketing e desempenha um papel fundamental na promoção do *site*.

5.6.3.Google Adwords

O Google Adwords é um serviço de propaganda fornecido pela Google. Com ele, é possível promover as páginas de um *website* nas buscas feitas pelo Google, pois os *links* aparecem como "*links* patrocinados", dentre os resultados da busca. Neste serviço, é possível escolher parâmetros para restringir a propaganda, com o objetivo de que somente apareça a seu público alvo. Ou seja, pode-se escolher o país, idioma e palavras-chave como parâmetros para que os *links*

patrocinados apareçam ou não em uma determinada busca. Também é possível estabelecer um orçamento diário de gastos, uma vez que a cada *click* do usuário, é descontado da conta da organização inscrita. Assim, ao acabar o crédito do dia, a propaganda não aparece mais até o dia seguinte.

5.7.Resumo

Há um conjunto muito extenso de ferramentas disponíveis para o auxílio no desenvolvimento de software, porém é necessário saber como estas ferramentas se integram e prover o treinamento necessário ao pessoal que deve utilizá-las.

É importante a criação de uma infraestrutura que suporte a disseminação e armazenamento do conhecimento (como *wikis*, por exemplo) e que facilite o trabalho diário, como ambientes de desenvolvimento que integrem várias funcionalidades (edição, teste, controle de versão, etc).

Normalmente, as pessoas são resistentes a mudanças e se a utilização de novas ferramentas ou metodologias for considerada um empecilho ao seu trabalho, certamente irão abandoná-las na primeira oportunidade. Portanto, é necessário que as mudanças sejam feitas de maneira incremental para que os impactos relacionados sejam minimizados. Além disto, é desejável que se tenha mais de uma opção a escolher, assim, todas as pessoas da equipe podem opinar e utilizar o conjunto de ferramentas que melhor lhes convier.