UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

DELCIO NONATO ARAUJO DA SILVA

WebTestManager: ferramenta de apoio ao Processo de Teste de Aplicações *Web*

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação

Orientadora: Profa. Dra. Ana Maria Alencar Price

Porto Alegre

2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Silva, Delcio Nonato Araujo da
WebTestManager: ferramenta de apoio ao Processo de Teste de Aplicações Web / por Delcio Nonato Araujo da Siva. – Porto Alegre: PPGC da UFRGS, 2003.
108 p.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2003. Orientadora: Price, Ana Maria de Alencar.

1. Engenharia Web. 2. Teste de Software. 3. Teste Aplicações Web. 4. Gerenciamento de Teste. 5. Qualidade de Software Web. I. Price, Ana Maria de Alencar. II. Título.

Dedico este trabalho a minha filha Lílian
e aos meus pais Nonato e Fátima.

Agradecimentos

A DEUS por ter me dado a saúde para a realização desta dissertação e ao curso de mestrado.

Aos meus pais pelas palavras de força e incentivo e por me ouvir nos momentos difíceis. Obrigado meu pai e obrigado minha mãe.

A minha filha e minha esposa pela paciência durante a realização da dissertação.

Aos meus irmãos pela colaboração e ajuda em todos os momentos da minha vida e especialmente na conclusão desta etapa.

Meus agradecimentos especiais a Profa. Dra. Ana Price pela orientação, apoio e compreensão dedicadas no decorrer do trabalho.

Ao CNPq pela ajuda financeira para a conclusão desta dissertação.

Aos professores e funcionários da Pós-Graduação do Instituto de Informática da UFRGS.

Aos professores do Departamento de Informática da UFPA, Arnaldo, Alfredo, Rodrigo, Carla, Sampaio, Cleidson e aos funcionários Mariano e Ivete.

Aos colegas do mestrado Cláudio, Alécio, Clóvis, Carolina, Rita, Olinda, Iraçú, Vanner, Paulo, Fábio pela companhia nos momentos de descontração e trabalho. E também aos amigos Lincoln, Eustáquio e Bruno.

TITLE: "WEBTESTMANEGER: TOOL FOR SUPPORT PROCESS WEB APPLICATION"

Abstract

This work presents a tool for the managing the testing process of Web based applications, which prioritizes the planning and execution phases with the integration of testing results from automated tools. In the last years, the World Wide Web has presented an extraordinary growth related to new applications on several areas, such as electronic trade, government services, education, and entertainment among others, needing though a larger quality control of applications based on that plataform.

Web Engineering is a new discipline whose objective is the use of processes, systematic approaches, administration and engineering principles with the purpose of designing, implementing, testing and maintaining high quality systems based on the Web. The quality and reliability of Web applications should be controlled on every software product. However, some inherent characteristics show that the Web applications all demand a larger concern because of the heterogeneity of both hardware and software platforms, and the great number of users.

The testing of Web applications all include several approaches such as validation of code, navigation, usability, safety, compatibility, functionality, interoperability, and integrity of data.

The tool proposed in this work, called WebTestManager, accomplishes the planning of the testing process for Web applications. Through it, test cases can be introduced in agreement with requirements for each testing type. This tool shall allow the storage of testing information and its results, intented, for metric evaluation of the performed test. That information can be obtained through the execution of automated tools for each testing type. As case studies, the tool is exemplified through a test planning for both a bank application and a flight ticket reservation application on the Web.

Keywords: Web Engineering, Software Test, Web Testing Application, Management of Test, Quality of Web Software.

Resumo

Este trabalho apresenta uma ferramenta para o gerenciamento do processo de teste de aplicações baseadas na Web, priorizando as fases de planejamento e execução através da integração com os resultados de ferramentas automatizadas. Nos últimos anos a World Wide Web apresentou um crescimento extraordinário com novas aplicações em diversas áreas, como por exemplo, comércio eletrônico, serviços governamentais, educação, entretenimento, entre outras, necessitando assim um maior controle de qualidade das aplicações baseadas na Web.

A Engenharia para Web é uma nova disciplina cujo objetivo é a utilização de processos, abordagens sistemáticas, princípios de gerenciamento e de engenharia com a finalidade de projetar, implementar, testar e manter sistemas e aplicações baseados na Web com alta qualidade. A qualidade e a confiabilidade das aplicações Web devem ser controladas como em todo produto de software. Porém, algumas características particulares mostram que as aplicações Web devem exigir uma maior preocupação, em função da heterogeneidade de plataformas de hardware e de software, e do grande número de usuários.

O teste de aplicações web deve abranger diversas áreas como validação de códigos, navegação, desempenho, usabilidade, segurança, compatibilidade, funcionalidade, interoperabilidade, confiabilidade e integridade dos dados.

A ferramenta proposta neste trabalho, chamada de WebTestManager, realiza o planejamento do processo de teste voltado para aplicações Web, na qual casos de testes são introduzidos de acordo com os requisitos de cada área de teste. Esta ferramenta permite o armazenamento de informações de teste e seus resultados, possibilitando uma avaliação através de métricas de teste. Essas informações podem ser obtidas através da execução de ferramentas automatizadas para cada tipo de teste. Como estudo de caso, a ferramenta é exemplificada no planejamento do teste de uma aplicação de bancária desenvolvida para a Web e de uma aplicação de reserva e compra de passagens aéreas.

Palavras-Chave: Engenharia Web, Teste de Software, Teste Aplicações Web, Gerenciamento de Teste, Qualidade de Software Web.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ASP	Active Server Pages
C/R	Capture / Replay
CGI	Common Gateway Interface
CSS	Cascade Style Sheet
CGI	Commom Gateway Interface
ENG	Engineering
FTP	File Transfer Protocol
GUI	Graphical User Interface
GUI	Graphics User Interface
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transfer Protocol
IIS	Internet Information Server
ISP	Internet Service Provider
JDBC	Java Database Connectivity
JSP	Java Server Pages
ODBC	Open Database Connectivity
OS	Operating System
PHP	Personal Home Page
RSI	Requirement-Service Interface
SQA	Software Quality Assurance
SQL	Structured Query Language
TCP/IP	Transfer Control Protocol / Internet Protocol
URL	Uniform Resource Locator
WWW	World Wide Web
XML	Extensible Markup Language

Sumário

1. Introdução

A Internet foi uma das grandes invenções das últimas décadas do século XX. O avanço com que a grande rede mundial de computadores trouxe para a área de comunicação, permitiu que informações fossem compartilhadas por milhares de usuário em todo mundo. O crescimento extraordinário mostrado pelo número de usuários, empresas e organizações que passaram a utilizar a Internet como meio para gerar, obter e compartilhar informações, prova a sua grande importância na vida moderna. O número de usuários estimado cresce dos 97 milhões no final de 1998 para 390 milhões no final de 2003, uma taxa de aumento anual em torno de 35%[1].

A principal aplicação na Internet é a World Wide Web, o qual facilitou utilizar e difundir os serviços disponíveis na grande rede. A Web surgiu com o objetivo de compartilhar documentos textuais entre pesquisadores. Porém, nos últimos anos observa-se o crescimento da utilização de vários tipos de tecnologias inserido em páginas Web, como vídeos, som entre outros. A WWW consolidou-se como novo formato de mídia em massa, tendo como uma característica o poder de interação entre os usuários e entre as empresas. A presença de uma empresa ou organização na Web acontece através de páginas Web, usualmente conhecido com Web Site.

Com o início do uso comercial, a Web passou a ser utilizada não somente em universidades e centros de pesquisa. Organizações e grandes empresas começaram a perceber os possíveis usos em benefícios dos seus negócios. Ho (1997), em sua avaliação para Web Site comerciais, classifica-os de acordo com objetivo do negócio em três categorias como: Promoção de Produtos e Serviços, Exibição de Dados e Informações e Transações e Processamento de Negócios.

Essa classificação apresenta como a Web pode ser utilizada de acordo com o grau de complexidade a ser implementada em Web Site, isto é, as várias partes de um grande Web Site usualmente permite a utilização de páginas Web simples e de forma estática enquanto outras podem ser geradas dinamicamente com a utilização de banco de dados. Segundo Ricca e Tornella (2001), a presença de páginas dinâmicas pode ser um critério para distinguir entre Web Sites (Estático) e Aplicações Web (dinâmico). Rocha (2001) mostra que a Web está deixando de ser orientada a documentos para ser orientada a aplicações.

Uma aplicação Web é “uma aplicação projetada para ser executada em um ambiente baseado na Web. Isto significa que uma aplicação Web não é somente coleções de páginas Web, mas a lógica da aplicação deverá ser integrada com a capacidade de hipermídia” (ATLANTIS et al., 2000). Desta forma é grande o número de aplicações que estão sendo desenvolvidas, migradas ou integradas com a Web, como forma de disponibilizar diversos serviços aos usuários.

Pressman (2000) aborda que o crescimento caótico das aplicações e sistemas baseado na Web lembra muito os primeiros dias da indústria de software. Não existia disciplina no desenvolvimento dos sistemas e a qualidade sempre era duvidosa. Assim, uma nova disciplina está surgindo com o objetivo de implementar conceitos de engenharia de software para a Web, chamada de Engenharia da Web.

A Engenharia da Web é uma nova disciplina que tem a preocupação com o processo de desenvolvimento de sistemas e aplicações baseado na Web. A essência da Engenharia da Web é gerenciar com sucesso a diversidade e a complexidade do desenvolvimento de aplicações web e desta forma evitar potenciais falhas que podem

[1] Estimado por *International Data Corporation* (www.idc.com)

provocar sérias implicações (GINIGE, 2001). Para Murugesan (1999), a Engenharia da Web tem como objetivo a utilização de processos, abordagens sistemáticas, princípios de gerenciamento e engenharia com a finalidade de projetar, implementar, testar e manter sistemas e aplicações baseados na Web com alta qualidade.

A fase de teste, verificação e validação têm como objetivo garantir o controle de qualidade das aplicações Web como em todo produto de software. Teste de software é definido por Binder (1999) como uma atividade na qual o sistema ou um componente é executado sobre uma especificada condição, e os resultados são observados, armazenados; uma avaliação é feita sob de algum aspecto do sistema ou componente. A fase de teste no desenvolvimento de aplicações web é tão importante quanto no processo de desenvolvimento de software tradicional.

Para que a fase de teste garanta a qualidade das aplicações Web, é necessário definir quais as principais características de qualidade. Powell (1998) lista um conjunto de qualidades semelhantes ao software tradicional, como: corretude, testabilidade, manutenbinilidade, portável, escalável, reusável, robusto, confiável, eficiente, boa documentação, apresentação apropriada e boa leitura. Vários autores definem outras características de qualidade que são apresentadas no capítulo 2. O importante é salientar que pela complexidade e utilização das aplicações Web, é necessário prover uma maior atenção para a área de teste (NGUYEN, 2001).

Diversos tipos de testes devem ser realizados com o objetivo de atender a todos os requisitos de qualidade. Dustin (2001), Splaine (2001), Nguyen (2001), Stottlemyer (2001) apresentam algumas abordagens práticas e concentradas nos objetos para testar e como realizar os testes. Freqüentemente, a lista de testes é estruturada em uma ou mais formas baseadas na experiência de desenvolvimento de cada autor. Dessa forma, cada um apresenta o que deve ser testado como sugestão, porém muitos tipos de testes são comuns e outros não. Os tipos de teste mais comuns voltados para aplicação Web são: usabilidade, navegação (*links*), acessibilidade, funcionalidade, garantia de código, compatibilidade, banco de dados, conteúdo, desempenho, escalabilidade, disponibilidade, confiabilidade, segurança.

Como realizado na fase de teste de software tradicional, a automação é de grande importância, pois muitos tipos de teste são praticamente difíceis, para afirmar quase impossível, de executar manualmente. Existem diversas ferramentas de teste voltadas para aplicações Web, como mostrado em Hower (2001), que possuem um objetivo específico de teste como: analisadores estáticos, detectores de erros dinâmicos, desempenho, segurança, compatibilidade, entre outras.

A utilização de algumas as ferramentas no processo de teste de aplicações apresenta uma grande dificuldade devido à falta de planejamento na utilização dos resultados e também no tempo que é necessário na execução para vários tipos de testes.

Desta forma, existe a necessidade de armazenamento dos resultados dos diversos tipos de testes com o objetivo de encontrar aspectos relevantes entre os resultados apresentados.

1.1. Objetivos

O presente trabalho tem como objetivo principal à modelagem e implementação de uma ferramenta com a finalidade planejar e gerenciar testes de aplicações Web. O módulo de planejamento permite a definição de requisitos dos testes, casos de testes e as versões de execução para cada tipo de teste voltado para aplicações Web. O módulo de

gerenciamento permite capturar informações dos testes realizados por ferramentas automatizadas as quais são responsáveis pela execução do teste.

A ferramenta permite armazenar os resultados com o objetivo de fazer um levantamento das métricas de testes baseado em aplicações Web. Assim, é possível analisar o comportamento da aplicação em relação a várias características de qualidade e também obter informações para ser utilizadas como critérios para novos testes em outras aplicações. Portanto, a ferramenta proposta pretende automatizar o planejamento para todos os tipos de teste e gerenciar as ferramentas que executam esses testes.

1.2. Estrutura

Este trabalho apresenta no Capítulo 2 uma visão das Aplicações de Software desenvolvida para a Web, abordando conceitos da área de Engenharia Web. Apresentam também componentes da arquitetura de aplicações Web e, finalmente, um estudo das características de qualidade de sistemas e aplicações baseada na Web.

No Capítulo 3, é apresentado o estado da arte em teste de aplicações Web. São abordadas as principais áreas de teste, abordando estratégias, métricas e a utilização de ferramentas automatizadas de teste. No final do Capítulo, são apresentados os benefícios e problemas da utilização de ferramentas automatizadas e uma avaliação com algumas ferramentas, considerando suas características essenciais, para utilizá-las neste trabalho e também as principais ferramentas por fornecedor.

O processo de teste voltado para aplicações Web é abordado no Capítulo 4. O estudo mostra as principais fases no processo de teste como forma de obter um melhor gerenciamento das atividades. São apresentadas também algumas considerações de ferramentas e suas características no gerenciamento do processo de tese.

A ferramenta proposta, WebTestManager, é descrita no Capítulo 5. A ferramenta aborda características no planejamento de teste de aplicações Web e apresenta como uma funcionalidade principal à integração com os resultados de outras ferramentas de testes específicos de aplicações Web. Essa integração permite um melhor controle dos principais testes a serem realizados em sistemas baseado na Web.

O Capítulo 6 apresenta um estudo de caso utilizando a ferramenta, abordando os resultados obtidos em função dos objetivos da aplicação.

Finalmente no Capítulo 7, é apresentada a conclusão deste trabalho e sugestões para trabalhos futuros.

2. Aplicações de Software Web

Em menos de uma década de existência, a utilização da Internet teve um crescimento extraordinário através da World Wide Web, diversas aplicações de software estão sendo disponibilizadas através de aplicações Web. Comércio, educação, mercado financeiro, saúde e governo são algumas áreas importantes que estão crescendo com aplicações voltadas para a Web. Isso mostra o alcance e o tamanho das atividades e aplicações baseadas na Web. Concentradas inicialmente na disseminação de informações, essas aplicações tornaram-se grandes aplicações de software na Internet, (Rocha, 2001) mostra que a Web deixou de ser "orientada a documentos" para ser "orientada a aplicação".

O crescimento caótico das aplicações e sistemas baseada na Web lembra em muito os primeiros dias da indústria de software (PRESSMAN, 2000). Não existia uma disciplina no desenvolvimento dos sistemas e a qualidade sempre era duvidosa. Assim, uma nova disciplina está surgindo com o objetivo de implementar conceitos de engenharia de software para a Web, chamada de Engenharia para Web. O objetivo é adotar princípios científicos, de engenharia e gerenciamento com uma abordagem sistemática para a obtenção de sucesso no desenvolvimento de aplicações de alta qualidade para sistemas baseados na Web.

Uma característica importante nas aplicações voltadas para a Web, é a complexidade da arquitetura. Dependendo do tamanho da aplicação, diversas categorias de equipamentos de hardware e de pacote de software são utilizadas para suportar a execução satisfatória das aplicações Web. Outras características são os aspectos de qualidade que devem ser verificados, validados e testados com o objetivo de garantir a qualidade das aplicações.

Este capítulo tem como objetivo apresentar um estudo de aplicações de software baseada na Web. A seção 2.1 apresenta os conceitos da área da Engenharia para Web. Na seção 2.2 é estudada a arquitetura utilizada em aplicações voltadas para Web. A seção 2.3 mostra características de qualidade das aplicações Web e finalmente a seção 2.4 apresenta tecnologias utilizadas no desenvolvimento de aplicações, tais como linguagens de programação, sistemas de banco de dados entre outros.

2.1. Engenharia para Web

Nos primórdios da indústria de software, o desenvolvimento era realizado sem nenhuma técnica formal ou alguma sistemática que apoiasse o processo na construção de software. Isso levou a crise do software (Pressman, 2000), a qual ocasionou diversos problemas no desenvolvimento e manutenção de software em operação e nas áreas críticas no mercado. Desta forma, surgiu a necessidade de utilização de princípios de Engenharia que garantissem a qualidade e confiabilidade dessas aplicações. Surgiu assim a disciplina de Engenharia de Software com o objetivo de aplicar princípios, métodos, metodologias e ferramentas automatizadas para o desenvolvimento de software.

Com o surgimento e o crescimento da Internet e da World Wide Web nos últimos anos, várias aplicações e sistemas de software estão sendo desenvolvidos baseados na Web, ou migrados e integrados com aplicações existentes. O escopo e a extensão das aplicações aumentaram de modo que áreas como educação, sistemas financeiros, governos, comércio eletrônico, entre outras, se transformaram em aplicações que oferecem serviços vitais para a sociedade. Desta forma, existe a

necessidade de que sistemas baseados na Web apresentem confiabilidade e desempenho e outras características como qualquer outro software tradicional (POWER, 1998). Portanto, devem possuir um processo de desenvolvimento que envolve as fases do ciclo de vida da aplicação. Fases como levantamento de requisitos, análise e projeto, implementação, teste e manutenção são algumas atividades que devem ser seguidas para o desenvolvimento de aplicações com qualidade.

Devido à falta de uma abordagem disciplinada para desenvolvimento de sistemas baseados na Web, Murugesan (1999) propôs uma nova disciplina, chamada de Engenharia para Web, que tem como objetivo a utilização de processos, abordagens sistemáticas, princípios de gerenciamento e de engenharia com a finalidade de projetar, implementar, testar e manter sistemas e aplicações baseados na web com alta qualidade. Desta forma, a Engenharia para Web se encontra cada vez mais presente no desenvolvimento de novas aplicações.

A Engenharia para Web é também composta por diversas áreas, pois a própria natureza das aplicações exige. Power (1998) descreve que sistemas baseados na Web envolvem uma combinação entre desenvolvimento de software e publicação de documentos, entre computação e marketing, entre comunicação interna e relações externas, e finalmente entre arte e tecnologia. Assim, o desenvolvimento de aplicações complexas exige conhecimento de diferentes áreas e também de profissionais especializados. A Figura 2.1 apresenta a Engenharia para Web como um campo multidisciplinar (MURUGESAN, 1999).

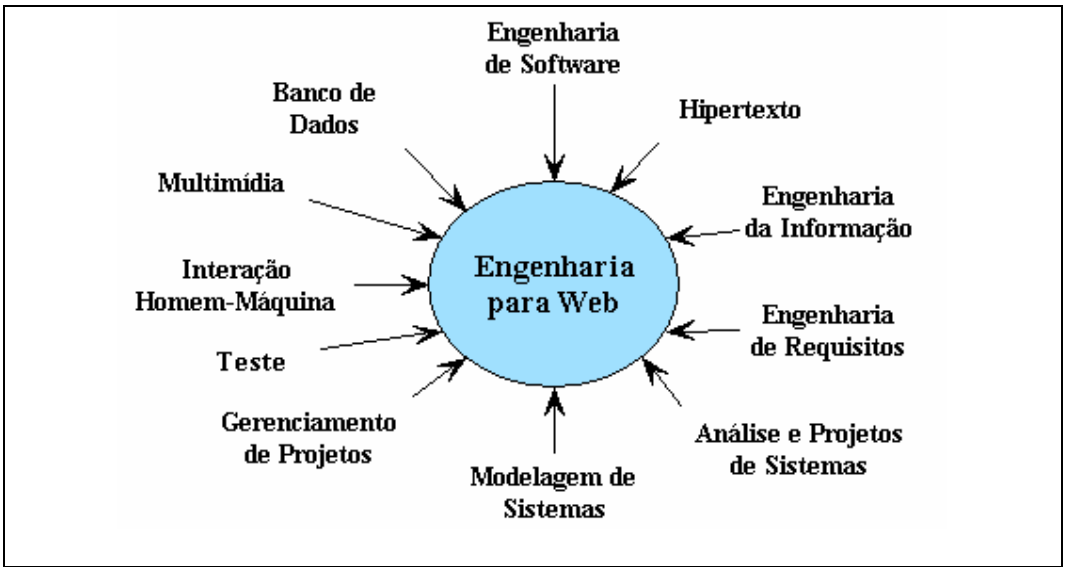


FIGURA 2.1 - Áreas da Engenharia Web

Como qualquer software, o sucesso de desenvolvimento e operação de uma aplicação Web está no processo de desenvolvimento. Engenharia Web possui uma abordagem com todos os aspectos de desenvolvimento (GINIGE, 2001), iniciando desde a concepção, passando pelo desenvolvimento até a sua implementação, teste e manutenção. Algumas das atividades realizadas na Engenharia para Web são apresentadas a seguir, segundo (MURUGESAN, 1999):

- Análise e Especificação de Requisitos;
- Técnicas e Metodologias de desenvolvimento de sistemas Web;

- Integração com sistemas legados;
- Migração de sistemas legados para ambientes Web;
- Desenvolvimento de aplicações em tempo real baseada na Web;
- Validação, Verificação e Teste;
- Garantia e controle de Qualidade;
- Métricas Web para estimativa e esforço de desenvolvimento;
- Avaliação e Especificação de Desempenho;
- Manutenção e Atualização;
- Aspectos Culturais e Humanos;
- Educação e Treinamento.

Ginige (2001) mostra que, ao contrário do que muitos profissionais da Engenharia de Software e desenvolvedores pensam, a Engenharia para Web não é um clone da Engenharia de Software apesar de que ambas envolvem programação e desenvolvimento de software. Ainda segundo Ginige (2001), enquanto Engenharia para Web adota e incorpora muitos princípios da Engenharia de Software, ela também incorpora novas abordagens, metodologias, ferramentas, técnicas e normas para encontrar requisitos únicos de sistemas baseados na Web. Murugesan (1999) apresenta algumas diferenças entre a Engenharia para Web e Engenharia de Software:

1. Sistemas Web são orientados a documentos com páginas Web com conteúdo estático ou dinâmico;
2. Aplicações Web são focalizadas na aparência, isto é, na visualização e apresentação das interfaces;
3. Ênfase no desenvolvimento e apresentação de conteúdo;
4. Utilização de múltiplos perfis de usuários;
5. Sistemas Web são desenvolvidos em um curto período de tempo, apresentando dificuldades em aplicar um planejamento formal de teste como usado no desenvolvimento de software tradicional;
6. Desenvolvedores de aplicações Web devem apresentar uma grande variedade de conhecimento de tecnologias e percepção quanto à qualidade de sistemas baseados na Web.

A Web está presente atualmente também em diversas áreas de negócios entre empresas. Sistemas de informações gerenciais ou de apoio à decisão nas organizações estão cada vez mais migrando para plataforma Web com o objetivo de melhorar as transações seja através da Internet, Intranet ou Extranet (HENDRICKSON, 2002).

Em Deshpande (2001), a Engenharia para Web é uma área que adiciona aspectos de Ciência da Computação, Sistemas de Informação e Engenharia de software. Essa integração de áreas de dar pelo fato que essas áreas apresentam as mesmas especificações comuns para o desenvolvimento de aplicações, como: processamento lógico, gerenciamento de dados e interface com usuário. Desta forma, profissionais da área de tecnologia da informação devem possuir conhecimento de diversas áreas, como mostrado na Figura 2.1, para o desenvolvimento de aplicações baseados na Web.

2.2. Aplicações Web e Web Sites

Segundo Nguyen (2001), as aplicações *web* são mais complexas que os sistemas tradicionais por apresentarem grande número de componentes, tanto de *software* quanto de *hardware*, os quais devem ser compatíveis como protocolos de redes, sistemas operacionais, navegadores (*browsers*), sistemas de segurança entre muitos outros.

TABELA 2.1 - Categorias de Aplicações Web

Categorias	***Exemplos***
Informacional	Jornais *on-line*, catálogos de produtos, revistas, manuais de serviço, classificados *on-line*, livros eletrônico *on-line*.
Interatividade	Formulários para registro, apresentação de apresentação customizada, jogos *on-line*.
Transacional	Comércio Eletrônico, Bancos *on-line*, registro de serviço *on-line*.
Workflow	Sistemas de planejamento *on-line*, gerenciamento e monitoramento de status de sistemas.
Ambientes de Trabalho Colaborativo	Ferramentas de projeto colaborativo
Mercados e Comunidade *On-line*	Grupos de *Chat*, supermercados *on-line*
Portais Web	Conjunto de serviços da internet

A Tabela 2.1 apresenta algumas categorias e os exemplos de aplicações de *software* para *Web*. Cada categoria possui sua particularidade quanto à forma de apresentação e interação do conteúdo. Para cada tipo de categoria é exigido um grau de qualidade da aplicação. Por exemplo, aplicações que envolvem a interação com o usuário através de prestação de serviços devem possui uma confiabilidade maior. A Tabela 2.2 mostra a diferença de uma aplicação *Web* simples e avançadas. Os aspectos de uma aplicação *Web* avançada indicam a utilização de técnicas e metodologias no desenvolvimento com o objetivo de obter aplicações confiáveis e de qualidade.

TABELA 2.2 - Características de Aplicações Web

Aplicações Web Simples	***Aplicações Web Avançados***
Páginas Web simples apresentando informações textuais.	Complexas Páginas Web.
Conteúdo de informação não muda, conteúdo estático.	Informação é dinâmica – mudanças ocorrem quando o usuário necessita.
Navegação Simples.	Dificuldade de navegação e busca de informação.
Sistemas Isolados.	Sistemas integrados com banco de dados.
Alto desempenho não é a maior requisito.	Requer desempenho alto e contínua disponibilidade.
Desenvolvimento por uma equipe pequena ou por somente profissionais individuais.	Requer uma equipe grande com especialistas em diversas áreas.

Power (1998) apresenta uma classificação baseada no grau de interação que uma aplicação *Web* ou *Web site* oferece:

1. *Web Sites* Estáticos – apresentam somente documentos em HTML. A interação é oferecida somente pela escolha dos *links* para navegar entre as páginas *web*.
2. Interação baseada em Formulários Estáticos – Formulários são usados para coletar informações dos usuários ou requisições para informações. O objetivo principal é a entrega de documentos, limitando somente a mecanismos de coleção de dados.
3. *Sites* com acesso a dados dinâmicos – *web sites* são usados como meio de acesso a base de dados. Por uma página *web*, usuários podem realizar pesquisas e/ou consultas a um conteúdo de um banco de dados. O resultado é mostrado como documentos HTML.
4. *Sites* gerados dinamicamente – o desejo, neste caso, é a geração de páginas customizadas para todos os usuários a partir de uma base de dados. A apresentação pode ser estática, não provendo interação, porém a forma como foi criada se assemelha com aplicações de *software*.
5. Aplicações de *Software* baseadas na *Web* – *web sites* que são parte de um processo de negócio geralmente tem mais em comum com outras aplicações cliente/servidor do que *web sites* estáticos.

A Figura 2.2 apresenta essa classificação quanto ao tipo da aplicação e o grau de complexidade. À medida que aplicações utilizam mais tecnologias, como banco de dados, mais complexidade é exigida da aplicação.

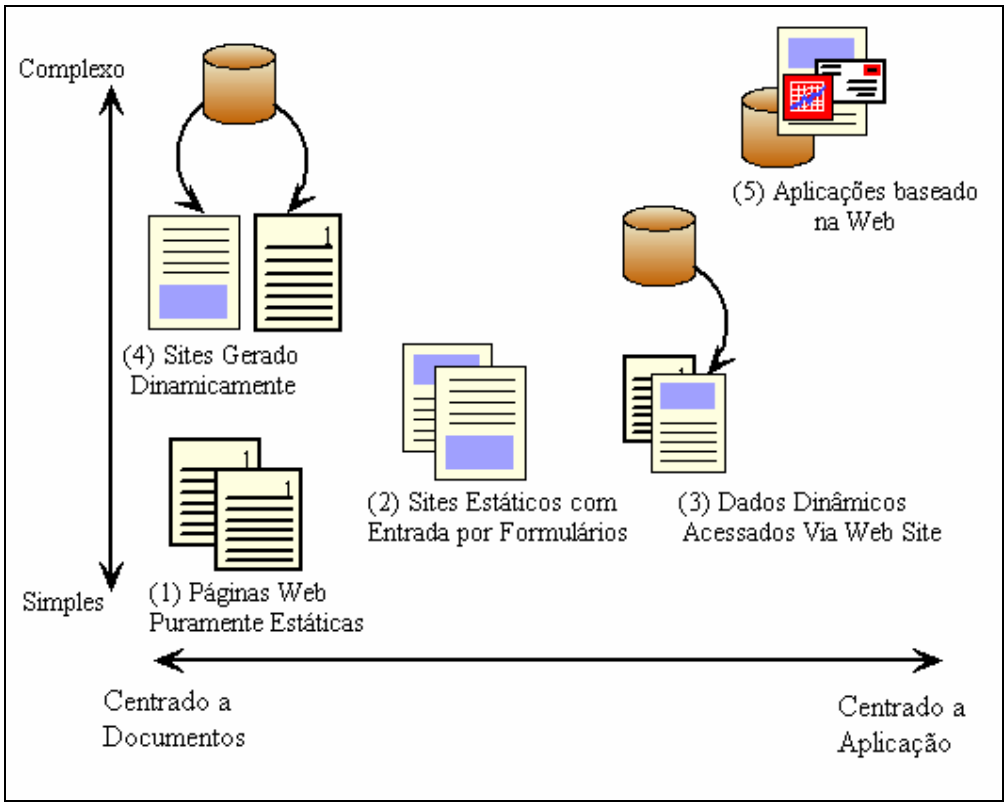


FIGURA 2.2 - Classificação de Sites (Powell, 1998)

Segundo (Power, 1998), existe uma diferença entre web sites e aplicações web. Os Web Sites são considerados como um conjunto de documentos ligados a outros documentos, cujos conteúdos são totalmente estáticos. Por outro lado, aplicações web podem obter dados e oferecer documentos de forma dinâmica através de formulários de entrada de dados, acesso a dados dinamicamente ou a geração de documentos dinâmicos utilizando banco de dados. As Aplicações web são consideradas mais complexas por envolver processos lógicos da aplicação, como a manipulação banco de dados. Uma aplicação web é definida como uma aplicação que o usuário pode trabalhar com conteúdos dinâmicos e afetar a lógica em um servidor, isto é, permitir que o usuário mude o estado lógico no servidor através de um software navegador web (browsers) (COLLINS, 2002).

As Aplicações Web podem ser consideradas como software (AMBLER, 2002) por apresentarem características que são inerentes ao software tradicional. Portanto, devem possuir um processo de desenvolvimento que envolve as fases do ciclo de vida da aplicação. Fases como levantamento de requisitos, análise e projeto, implementação, teste e manutenção são algumas que devem ser seguidas para o desenvolvimento de aplicações com qualidade.

As Aplicações web devem apresentar características semelhantes, quanto à qualidade, ao software tradicional como ser eficientes, corretos, reusáveis, robustas, confiáveis, portáveis, manuteníveis e documentadas. Uma grande diferença está no tempo de desenvolvimento e atualização. Enquanto o software tradicional leva alguns meses para ser desenvolvido e também algum tempo para a manutenção, as aplicações web são construídas em um pequeno espaço de tempo e com uma manutenção muito rápida.

2.3. Arquitetura de Software Web

Uma arquitetura de aplicação web pode ser visualizada como um modelo de uma aplicação tradicional de transação de negócios, tendo como diferença um modelo cliente/servidor no qual está baseada toda a tecnologia utilizada por aplicações web. Um sistema web consiste de muitos clientes e servidores os quais possuem diversos componentes de hardware e aplicações de software. Um cliente faz requisições para um servidor que processa os dados e entrega o resultado de volta ao cliente, conforme mostrado na Figura 2.3. A comunicação entre o cliente e o servidor é realizada através de uma rede, que pode ser local ou de longa distância. Assim como sistemas cliente/servidor, os sistemas web são aplicações de acesso aos dados com a particularidade de utilização de navegadores (browsers) para o acesso aos servidores.

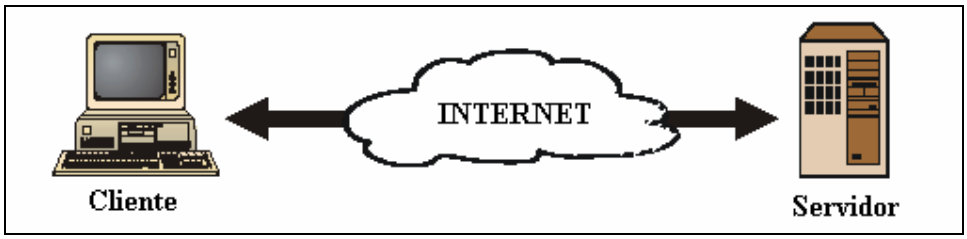


FIGURA 2.3 - Arquitetura do Modelo Cliente-Servidor

A Figura 2.4 apresenta a arquitetura de um sistema Web, que consiste de um grupo de computadores interconectados através da Internet que suporta a recuperação de informações e a implementações de serviços. A arquitetura é distribuída em três ou mais

camadas: camada de apresentação, camada de conteúdo e aplicação e a camada de gerenciamento de dados e serviços. Todas as camadas são ligadas e suportadas por equipamentos de hardware e uma infra-estrutura de rede. Sistemas Web de menor porte podem possuir algumas ou todas as camadas em somente um computador. Em contraste, grandes sistemas Web podem requerer mais camadas para suportar o processamento, integração com banco de dados existentes e serviços de apoio.

Essa complexidade de tecnologia de hardware e software torna aplicações Web um ambiente muito vulnerável a erros, mostrando assim os riscos na implementação. Outra questão é a quanto à diversidade dos produtos de hardware e software de diferentes fabricantes que devem interagir entre si, tornando a arquitetura com conteúdo mais heterogêneo e com maior probabilidade de ocorrência de falhas.

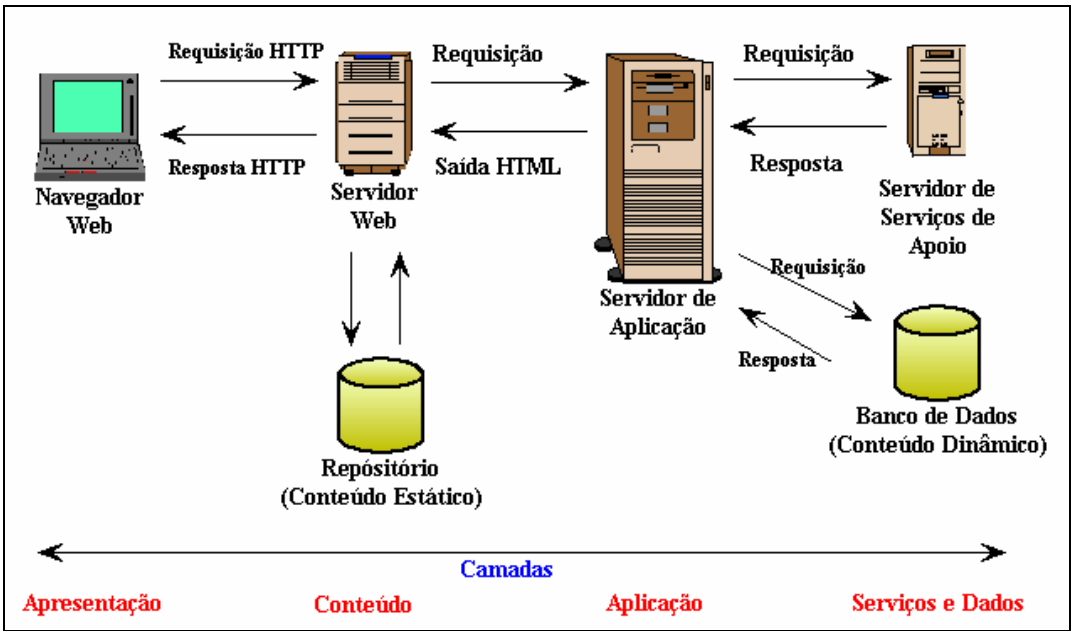


FIGURA 2.4 - Arquitetura Completa de Aplicações Web

Muitos sistemas Web podem ser construídos usando a abordagem de n camadas, permitindo uma excelente escalabilidade, assim como ótima distribuição de atividades das aplicações entre as camadas de apresentação (cliente), aplicação, e banco de dados.

2.3.1. Evolução das Arquiteturas Web

Offut (2002) apresenta uma evolução das arquiteturas de aplicações Web até chegar a arquitetura apresentada na Figura 2.4. A primeira geração, por volta de 1993, apresentava documentos com poucas imagens, formato bem simples, páginas sem cores de fundo e programação muito limitada com formulários. Um profissional era responsável para desenvolver todo o site, existia pouca preocupação com projeto e geralmente ficava incompleto.

Por volta de 1994, quando web iniciou comercialmente, a segunda geração apresenta Web sites mais focado para o formato das páginas, cores e tabelas. Profissionais da área gráfica começaram a desenvolver páginas web, o que tornou uma maior ênfase na aparência devendo o sucesso à criatividade e técnica empregada, pessoas já não liam o Web site, já era realizado uma visita com algum objetivo.

Com a terceira geração, buscou-se mais interatividade com o usuário e novas tecnologias sugiram como Java, linguagens de Scripts, controles Active-X. A utilização quadro (frames), marcadores (tags) proprietários e folhas de estilos (CSS), também ajudaram a modificar o formato e a interação dos Web sites. Assim, cresceu em muitas aplicações web que exigiam uma melhor interação com usuário como: comércio eletrônico, serviços de reservas de vôo, entre outros.

Finalmente, a quarta geração, o qual a Figura 2.4 representa, mostra Web sites mais voltado para utilização como software, com muito mais complexidade e tecnologias utilizadas.

2.3.2. Camada de Apresentação

A camada de apresentação ou cliente é responsável pelos componentes de aplicação para o usuário final, consiste essencialmente de um software navegador (browser), como o Microsoft Internet Explorer ou o Nestcape Navigator. Os maiores navegadores suportam extensões para proverem funcionalidade adicionais, como plug-ins, applets e scripting. Esses componentes provêem uma interação com o usuário para capturar dados de entrada, disponibilizar dados de saídas, e potencialmente realizar algum tipo de processamento. Essa camada se comunica com a camada de aplicação através do Protocolo de Transferência de Hipertexto (HTTTP) ou Protocolo de Transferência de Hipertexto Seguro (HTTPS).

Nguyen (2001) define dois tipos de clientes de acordo com o tipo de processamento. Cliente Thin existe pouco processamento na camada do cliente, regras da lógica de negócios são executadas na camada de aplicação, isto é, no servidor. Esta abordagem centraliza processamento no servidor e elimina muitos problemas de incompatibilidade no lado cliente. O outro tipo é o Cliente Thick, o qual possui poder de processamento compartilhado com o servidor para a execução de componentes para funcionalidades adicionais.

A seguir são brevemente descritas algumas das tecnologias mais comuns que existe no lado cliente.

2.3.1.1 HTML/XHTML

A Linguagem de Marcação de Hipertexto (Hypertexto Markup Language, HTML) é a linguagem para mostrar páginas Web em um navegador. HTML provê um conjunto de marcadores (tags) para a separação de documentos Web em seções, visualizando dados em tabelas, áudio embutido e outros recursos. A ligação entre os documentos HTML é realizada através dos links. O XHTML é uma extensão da linguagem HTML, permitindo uma separação entre o conteúdo e a aparência dos documentos. Esta separação serve para o acesso de informações na Web por dispositivos diferente do computador, como o celular.

2.3.1.2 CSS (Cascade Style Sheet)

As folhas de estilo permitem um desenvolvedor Web especificar a posição, tamanho e outros atributos de um documento. CSS resolve problemas de compatibilidade com alguns navegadores que não suportam novas tags adicionadas em novas versões do HTML.

2.3.1.3 Linguagens de Scripts

Uma das formas mais comuns que uma página web interage com o usuário é através do uso de scripts, que permitem escrever código que executa no computador do cliente e interage com elementos da página web. Essa interação é habilitada pelo navegador, o qual expõe elementos das páginas e funções do navegador com elementos de scripts.

2.3.1.4 XML/XSLT

A HTML ou XHTML permite somente a distribuição de documentos e não de dados puros. XML, Linguagem de Marcação Extensiva, é a linguagem para trabalhar com dados através de estruturas que permitem a troca de informações através de marcadores (tags) personalizadas. Já XSTL, Linguagem de Transformação de Folhas de Estilo Extensiva, é usada para converter dados em XML para outro formato como HTML ou XHTML.

2.3.1.5 Plug-ins

A utilização de Plug-ins tem como objetivo a visualização de documentos que não seja em formato HTML. Desta forma, o navegador do usuário estende a capacidade de utilização para certos tipos de documentos.

2.3.1.5 Controle Active-X

Similares ao Plug-ins, controles Active-X são componentes que provêem um alto nível de funcionalidade no computador do cliente. Controles Active-X, embutidos dentro de páginas Web, pode ser utilizado para mostrar conteúdo especializado, como áudio de alta qualidade ou interação avançada com o usuário.

2.3.3. Camada de Aplicação ou Camada do Servidor

A camada de aplicação provê o processamento de sistemas e de negócios da aplicação. Serviços de aplicações são implementados entre os vários componentes da camada de aplicação. O servidor web é o principal componente da camada de aplicação, ele interage com clientes com ciclos de requisição e resposta. Um servidor pode processar uma requisição de várias formas. O cliente pode requisitar uma simples página web estática, a qual o servidor deverá ler do disco e enviá-la. Em outro caso, o cliente pode requisitar uma página dinâmica, requerendo o servidor ler e, potencialmente, executar código contendo na página e retornar a resposta ao cliente. A seguir, algumas das tecnologias para componentes da camada de aplicação.

2.3.3.1 Programas CGI

Programas CGI foram os primeiros mecanismos para conteúdo dinâmico na *web*. O Commom Gateway Interface (CGI) é um programa que é chamado pelo servidor *web* em resposta à requisição do cliente. Programas CGI podem ser escritos em diversas linguagens, tendo o servidor que informar alguns parâmetros para executá-los. Uma desvantagem é a ineficiência para aplicações grandes com alto volume de dados.

2.3.3.2 Páginas Dinâmicas

Em adição às páginas estáticas, servidores *web* têm a habilidade de entregar páginas dinâmicas, as quais podem recuperar dados ou realizar outras operações retornadas ao usuário. Exemplos de páginas dinâmicas são Microsoft Active Server Page (ASP), Java Server Pages (JSP) e Hypertext Pages (PHP). Essas tecnologias são muito mais eficientes do que programas CGI.

2.3.4. Camada de Dados e de Serviço

A camada de banco de dados fornece o gerenciamento de dados e serviços para a aplicação. Esta camada é responsável pela confiabilidade, escalabilidade e processamento de transação de dados da aplicação de alto volume. Esta camada armazena o sistema de gerência de banco de dados, arquivo de dados e qualquer componente de processamento usado para suportar transação com dados.

2.4. Qualidade de Software Web

A complexidade de uma aplicação web é grande devido à diversidade de componentes de hardware e software (Offut, 2002), como mostrado na seção anterior. Outra grande questão é quanto aos diferentes objetivos que um Web site ou uma aplicação web está focada (Rocha, 2001). O processo de garantia de qualidade exige a consideração de diversos aspectos que abrangem desde da qualidade da apresentação, como a disponibilização de conteúdo até a segurança de acesso.

Olsina (1999), Pressman (2000), Dustin (2001) e ATZENI (2001) apresentam algumas características de qualidade de aplicações web, como mostra a Tabela 2.3.

TABELA 2.3 - Qualidade em Aplicações Web

Aspectos de Qualidade	***Características Principais***
Usabilidade	Entendimento Global do Site
	Feedback e help on-line
	Características estéticas e de interface
	Características especiais
Funcionalidade	Capacidade de busca e recuperação
	Características de navegação
	Características relacionadas ao domínio da aplicação
	Processamento correto de *links*
Confiabilidade	Recuperação de Erros
	Validação e recuperação da entrada de usuário
Eficiência	Desempenho em tempo de resposta
	Velocidade de geração de páginas
	Velocidade de geração de gráficos
Manutenilibilidade	Facilidade de Correção
	Adaptabilidade
	Extensibilidade

2.4.1. Funcionalidade

Um fator principal de qualidade segundo os usuários é a funcionalidade (DUSTIN, 2001). Os requisitos definidos para a aplicação web deverão ser oferecidos pelos componentes da aplicação e pelas páginas web. As aplicações web apresentam inúmeras funcionalidades implementadas com forma de facilitar o uso por parte do usuário, tanto funções relativas ao domínio da aplicação como funcionalidades inerentes a sistemas baseados na web como navegação entre as páginas.

Um dos maiores problemas com a funcionalidade são a compatibilidade com os diferentes tipos e versões de navegadores e sistemas operacionais disponíveis no mercado. Em alguns casos, fica impossível utilizar a aplicação devido a problemas apresentados na máquina do cliente em relação à configuração dos principais navegadores (STOTTLEMYER, 2001).

Os componentes da interface gráfica com o usuário devem apresentar as suas respectivas funcionalidades no contexto da aplicação. Por exemplo, caixas de listagens devem selecionar os itens corretos, também verificar se caixas de texto recebem dados com os seus respectivos tipos de dados, etc. As funcionalidades dos componentes devem ser testadas quando utilizados quando da combinação entre os componentes.

2.4.2. Segurança

Um dos atributos relevantes para aplicações web é segurança, informações e dados de que trafegam entre aplicações e usuários em transações devem ser suportados por mecanismos que resistem ao acesso de pessoas não autorizadas. O fator de qualidade segurança deve ser bem planejado para aplicações críticas que envolvam informações sigilosas, como senhas de contas bancárias, pois qualquer problema apresentado torna a aplicação web sem credibilidade perante os usuários.

A questão de segurança é um grande desafio, pois pela própria natureza e arquitetura aberta da Internet, qualquer usuário pode conectar-se com uma aplicação sem autorização. Muitos sistemas operacionais e serviços da própria Internet incluem mecanismos de auditoria para detectar o acesso de usuários que não possuam permissão de acesso, porém não é uma atividade muito fácil devido ao surgimento a cada dia de novas formas de intrusão a sistemas baseados na Internet (DUSTIN, 2001).

2.4.3. Confiabilidade e Disponibilidade

A questão de confiabilidade está bem ligada com o sucesso, principalmente, de aplicações comerciais. Se uma aplicação não trabalhar como o esperado em qualquer tipo de transação, existe uma grande probabilidade de o usuário não retornar a utilizar a aplicação. Isso significa perda de receita e a imagem da empresa fica desgastada. Empresas que desejam manter aplicações na web devem prover recursos que suportem a confiabilidade ou não terão sucesso com a aplicação (OFFUT, 2002).

Além do mais, aplicações web são utilizadas por milhares de usuários que tem uma expectativa em relação ao resultado apresentado pela aplicação. Porém, em grande parte das aplicações, em algum momento do funcionamento já houve problemas que trouxeram dúvidas quanto à confiabilidade da aplicação (SPLAINE, 2001).

A confiabilidade é geralmente medida pelo tempo médio entre falhas, pelo tempo médio entre a falha e a recuperação ou pelo número de quedas durante um certo intervalo por um período de tempo especificado. Por exemplo, algumas empresas podem mensurar a confiabilidade como o número de quedas não pode exceder à uma

hora por mês. Alternativamente, o grau de aceitação de falhas para um recurso particular pode ser especificado, por exemplo, em qualquer momento de tempo, não mais do que 2% do links podem estar quebrados. A definição de falhas de aceitação depende dos procedimentos de cada empresa.

Assim, aplicações de software web devem prover operação vinte e quatro horas por dia, sete dias por semana, 365 dias por ano (sistemas 24x7x365), devem estar disponíveis para o usuário a qualquer hora de forma que o usuário consiga realizar suas transações satisfatoriamente.

2.4.4. Desempenho

Um dos principais benefícios de aplicações baseadas na Web é a permissão de utilização de múltiplos usuários acessando os recursos do sistema simultaneamente através dos navegadores, isto é, múltiplos usuários podem requisitar diferentes serviços com diferentes características (NGUYEN, 2001). Em função do suporte a múltiplos usuários, a aplicação deve realizar cuidadosamente funções críticas durante os períodos de uso de grandes quantidades usuários.

Diversos fatores estão relacionados com o desempenho de aplicações web, como a própria complexidade do ambiente e a característica dinâmica da Internet. Os problemas relacionados com o desempenho podem ocorrer em diversos pontos entre o usuário e o servidor web (SPLAINE, 2001).

Segundo Dustin (2001) o desempenho de sistemas web podem ser descritos de duas formas segundo, o tempo de resposta, para o usuário final e a utilização de recursos para o administrador dos sistemas. Esses dois fatores podem determinar o desempenho da aplicação como um todo, especificando assim o fator qualidade em relação à utilização de sistemas web que requerem uma melhor análise do desempenho.

2.4.5. Usabilidade

Usabilidade é a capacidade que um usuário de um web site tem de procurar o que ele necessita com um esforço razoável e por um período tempo satisfatório (SPALINE, 2001). A qualidade de uma aplicação web pode ser determinada também pelo grau de usabilidade proporcionada por todas páginas web da aplicação.

Em uma aplicação web, a relação com o usuário é realçada ou enfraquecida por fatores como uma fácil navegação, uma organização lógica de componente pela página web, um intuitivo fluxo de ações e mudanças de páginas, o uso de imagens e cores, a velocidade de respostas do sistema as ações do usuário e o valor do conteúdo mostrado na página web (DUSTIN, 2001).

2.5. Conclusões

Este capítulo apresentou uma visão geral da nova disciplina surgida com sistemas voltados para a Internet, a Engenharia para Web. O principal objetivo é o desenvolvimento de aplicações Web e Web Sites através de princípios que já são aplicáveis à Engenharia de Software tradicional, porém com algumas particularidades para aplicações baseadas na Web. O próprio ambiente dessas aplicações já se apresenta como um desafio para a Engenharia para Web, visto que são inúmeros as tecnologias empregadas e que devem ser integradas e também grandes números de novas aplicações que são demandadas a cada momento, sejam novos sistemas ou migração de sistemas antigos.

Desta forma, vários aspectos de qualidade devem levados em consideração para aplicações Web, como no software tradicional. Esses aspectos apresentam o grau de sucesso que uma aplicação Web poderá obter.

3. Teste de Aplicações Web

Como mencionado no capítulo anterior, aplicações web estão presentes cada vez mais nas empresas, organizações e, principalmente, através de serviços nas áreas de governo, educação, comércio eletrônico, bancos e portais de informações. Essas aplicações são sistemas 24x7, isto é, devem funcionar 24 horas por dia e 7 dias por semana, apresentando características de qualidade até então não disponíveis para o software tradicional (SHAH, 2001).

A qualidade e a confiabilidade das aplicações web deve ser controlada tal como em todo processo de desenvolvimento de software e também no produto final. Porém, algumas características particulares mostram que as aplicações web devem exigir uma maior preocupação, em função da heterogeneidade de ambientes e do grande número de usuários (NGUYEN, 2001). A fase de teste, verificação e validação têm como objetivo garantir o controle de qualidade das aplicações web.

Murugesan (1999) mostra que a atividade de teste de sistemas baseado na internet é uma tarefa importante no processo de desenvolvimento, porém pouca atenção tem sido dada por desenvolvedores web. Ainda é necessário desenvolver novas abordagens e técnicas para teste e avaliação de sistemas complexos baseado na web.

O teste de aplicações Web envolve inúmeras questões e muitos desenvolvedores têm dado baixa prioridade para essa atividade por considerar uma tarefa tediosa (Power, 1998). É interessante frisar, dependendo da aplicação, que erros e falhas nas aplicações podem apresentar conseqüências desastrosas. Como, por exemplo, sistemas bancários e financeiros. A confiabilidade da aplicação não será questionada pelo serviço oferecido na Web, porém erros freqüentes na aplicação tornam os serviços sem credibilidade perante os usuários, e conseqüentemente inviabilizando todo projeto da aplicação na Web.

Este capítulo aborda as principais estratégias de teste para aplicações web. A seção 3.1 apresenta os principais conceitos de teste de software. Na seção 3.2 são apresentadas as áreas que devem ser testadas em sistemas baseados na web e a seção 3.3 são apresentadas considerações sobre ferramentas automatizadas que auxiliam na tarefa de teste.

3.1. Principais Estratégias de Teste

A atividade de teste de software é uma das mais importantes no processo de desenvolvimento para a garantia da qualidade do software. Muitas equipes de desenvolvimento gastam um tempo considerável na fase de teste, cerca de 40% segundo Pressman (1995), o que mostra que esta etapa deve ser bem planejada. Em Hetzel (1987), o teste é um processo de aquisição de confiança no fato de que um programa ou sistema faz o que se espera dele. Já Myers (1979), o teste é o processo de executar um programa ou sistema com a finalidade de encontrar erros. Em geral, é impossível executar um teste de software completo, já que o teste não garante a ausência de erros, mas apenas revela a presença de defeitos.

Teste é a medida de garantia de qualidade que acompanha todo o projeto do software. Pomberger e Blaschek (1996) compararam a atividade teste como a garantia de qualidade do produto enquanto que a qualidade do processo é de responsabilidade da atividade de garantia de qualidade. Para Marick (2000) teste é um conjunto de ações necessárias de planejamento sistemático para dar confiança que produto ou serviço satisfaz um padrão de qualidade definido.

Segundo Pressman (2000), o crescente destaque do software como elemento dos sistemas de informação e os custos associados às falhas de software são forças propulsoras para uma atividade de teste cuidadosa e bem planejada. Um processo de teste deve fornecer um nível de confiança tal a fim de que o software encontre, através de condições específicas, seus objetivos e também deve detectar eventual falha nos mesmos. O cerne do teste estará na localização de erros no software, pois ao encontrá-los, tem-se a certeza que o teste foi bem sucedido.

3.1.1. Teste Funcional ou Caixa Preta

O teste funcional, conhecido também como teste de caixa preta (black Box), tem como objetivo validar as funções da aplicação de software tendo como base a especificação de requisitos do usuário. Os casos de teste são derivados das especificações funcionais que o programa deve realizar. As especificações definem como a aplicação deve comporta-se sob o ponto de vista externo, determinando as entradas e saídas e enumerando as funções realizadas. As técnicas orientadas para este tipo de teste derivam os dados de teste a partir da análise da funcionalidade do programa, sem levar em consideração a estrutura dos mesmos.

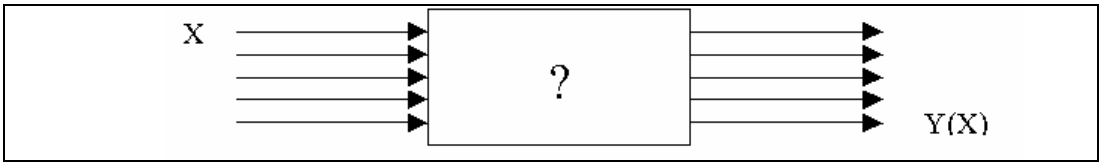


FIGURA 3.1 - Teste Funcional

A Figura 3.1 apresenta um exemplo de teste funcional, onde o código e a estrutura interna são desconhecidos. As categorias de erros, segundo Pressman (1995), mais evidenciadas são: erros de interface, erros de desempenho, funções incorretas ou ausentes, acesso à banco de dados externos e erros de inicialização e término. O teste funcional é geralmente aplicado quando todo ou quase todo o sistema já foi desenvolvido.

3.1.2. Teste Estrutural ou Caixa Branca

Conhecido como teste de caixa branca (white box), o teste estrutural deriva da lógica de programa e da estrutura de codificação, isto é, os casos de teste, são derivados a partir da análise da estrutura interna do programa. O código e a arquitetura são conhecidos do testador e os testes deverão ser projetados e executados de uma maneira que garantam uma cobertura completa do código, embora algumas áreas são menos importantes ou menos executadas durante a execução de uma aplicação. Desta maneira, os casos de teste têm como objetivo causar a execução de caminhos identificados no programa, baseado no fluxo de controle e/ou no fluxo de dados. Uma vez selecionados os caminhos que devem ser executados, os próximos passos envolvem a geração de dados que causam a execução destes caminhos. A Figura 3.2 apresenta um exemplo para o teste estrutural.

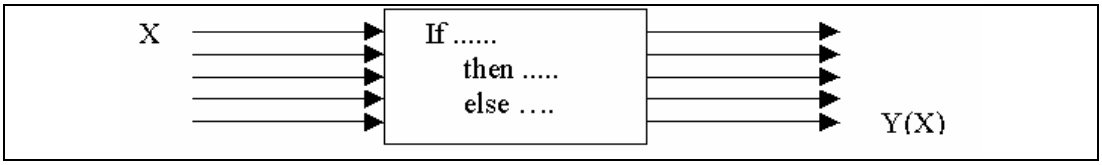


FIGURA 3.2 - Teste Estrutural

3.1.3. Teste de Caixa Cinza

Uma nova abordagem é o teste de caixa cinza (Gray Box) apresentado por Nguyen (2001) e Kaner (1999). O teste envolve tanto entradas e saídas e o conhecimento da lógica interna dos programas ou componente, ou seja, é uma combinação das duas abordagens definidas acima. Esse tipo de teste é mais utilizado em aplicações baseada na Web, pois apresenta uma abordagem de verificação funcional de vários componentes e como é o funcionamento interno de outros componentes e quais as condições de interoperabilidade. Estes componentes devem ser testados no contexto do projeto do sistema para avaliar a real funcionalidade e compatibilidade.

3.2. Áreas de Teste de Aplicações Web

Aplicações web compartilham muitas similaridades com aplicações tradicionais Cliente/Servidor, mas existe também um número de diferenças que criam novos problemas no momento da realização dos testes. Como primeiro exemplo, pode-se listar a grande variedade de tecnologias utilizadas pelos desenvolvedores (como mostrado no Capítulo 2). Outro grande problema é o número de combinações que as aplicações deverão executar no lado cliente. Sistemas Web deverão trabalhar com vários sistemas operacionais, navegadores, arquiteturas de computador pessoal entre outros (HIEALT; MEE, 2002)

Em aplicações Cliente/Servidor, o número de usuários que está utilizando o sistema é geralmente, fácil de mensurar. Porém para aplicações Web, essa tarefa apresenta muita dificuldade de ser realizada. É difícil conhecer o número de usuários que irá utilizar a aplicação e sua variação em um determinado dia ou hora.

Estes são alguns dos desafios apresentados no desenvolvimento e teste de aplicações Web. Outras questões do teste de aplicações Web são as dificuldades de solucionar os defeitos. As muitas camadas que interagem na arquitetura da aplicação podem ser responsáveis pelos erros ou apresentarem os sintomas dos erros ocorridos. Nguyen (2001) apresenta cinco considerações importantes:

- Quando o usuário visualiza um erro no lado Cliente, ele está vendo um sintoma do erro e não o erro em si;
- Erros podem ser dependentes do ambiente que a aplicação está sendo executada e podem não aparecer em diferentes ambientes;
- Erros podem ocorrer no código ou ser problemas de configuração;
- Erros podem residir em qualquer uma das várias camadas (Cliente/Aplicação/Servidor);
- As duas diferentes classes de ambientes operacionais, estático versus dinâmico, demandam diferentes abordagens.

Kallepalli (2001) define que a garantia de qualidade do teste de aplicações Web focaliza a prevenção de falhas ou a redução de chances de cada falha. Uma falha Web é definida como a inabilidade da entrega de forma correta da informação ou dos documentos requeridos por usuários Web. Baseado nesta definição podem ser consideradas as seguintes origens de falhas associadas com as diferentes camadas da arquitetura Web (apresentadas na Figura 2.4): falhas na rede ou servidor, falhas no navegador (Browser) e falhas de conteúdo ou origem no servidor.

As áreas a serem testadas são apresentadas de acordo com as diversas abordagens apresentadas em Nguyen (2000), Tongeren (1999), Soberano (2000), Powell (1998), Splaine (2001), Helm (2001), Shah (2001), Dustin (2001), Gerard (2000), Miller (2000) e Stout (2001). É importante salientar que existem algumas áreas que são iguais na visão dos mesmos autores e outras são divergentes. Desta forma, na Tabela 3.1 apresentada a seguir, buscou-se identificar as áreas e sub-áreas mais importantes para o teste de aplicações Web.

TABELA 3.1 - Áreas de Testes de Aplicações Web

Áreas	Sub-Áreas
Interface com Usuário	Apresentação dos Dados, Mensagens de Aviso e Erros, Navegação, Formulários, Formatação.
Compatibilidade e Configuração	Sistemas Operacionais, Navegadores, Conexões, Impressoras, Dispositivos de Multimídia, Servidores de Aplicação, Servidores de Banco de Dados e de Web.
Desempenho	Tempo de Resposta, Carga, *Stress*, Tráfego de Usuários, Volume de Dados, Uso Contínuo.
Segurança	Protocolos, *Firewalls*, Logins, Configuração de Diretórios.
Funcionalidade	Linguagens de Programação, Links, Cookies, Transações específicas da aplicação, Testes orientados a tarefa.
Conteúdo	Apresentação de Imagens, Textos com Gramática correta.
Banco de Dados	Integridade

Shah (2001) apresenta alguns critérios gerais para o teste de aplicações Web, os quais seguem o proposto no modelo de teste para o teste de software tradicional:

- Teste de Unidade de páginas, componentes individuais e formatos verificando se estão de acordo com as especificações;
- Teste dos Links, checar as ligações entre páginas locais e páginas remotas;
- Teste de Integração, na qual a aplicação web instalada no Servidor Web é testada para verificar se realiza a tarefa como especificada e não causa erro em outras partes da aplicação;
- Teste de Aceitação, verifica-se que a aplicação Web foi desenvolvida de acordo com as especificações dos requisitos do negócio.

Gerrard (2000) e Stout (2001) apresentam uma abordagem do teste de aplicações web da seguinte maneira: Testes Funcionais e Testes Não-Funcionais. O Teste

Funcional aborda testes com transações específicas das aplicações e testes com navegação das páginas. Os Testes Não Funcionais englobam testes que avaliam questões como o desempenho, configuração, segurança, escalabilidade, entre outros. Essa divisão permite definir critérios a serem estabelecidos para prioridade no processo de teste.

Dessa forma, testar aplicações Web e Web Sites requer um levantamento preciso dos requisitos de negócios a fim de aplicar os testes de acordo com o objetivo e complexidade da aplicação (GLASS, 2000). Por exemplo, para uma determinada aplicação, se o foco principal deve ser quanto ao desempenho, assim, deve se planejar a realização de testes que atendam os requisitos de utilização por diversos usuários simultaneamente. Já em outra aplicação, o objetivo crucial dos testes pode envolver a segurança das informações que trafegam pela aplicação, portanto, todo cuidado deve ser voltado para esse tipo de testes.

A maioria dos livros e artigos sobre teste em aplicações Web citados neste trabalho apresenta abordagens práticas de como testar e como realizar os testes. Freqüentemente, uma lista de atividades de teste é determinada, estruturada de uma ou outra forma, dependendo do número de páginas, explicando os detalhes, de como executar os testes individuais com base na explicação da boa prática no desenvolvimento de aplicações Web e o que deve ser evitado. Como diretrizes para o testador, são sugeridos, às vezes, alguns planos de teste, com o objetivo de explicar os tipos de testes. Uma estrutura operacional também é apresentada.

Esta seção apresenta uma avaliação dos tipos diferentes de testes abordados na bibliografia deste trabalho. Cada um dos testes tem um foco principal, embora os objetos de teste subjacentes e os métodos de teste aplicáveis podem se sobrepor. Aqui, os testes são apresentados sem uma ordem particular. Uma tentativa para agrupar os testes de acordo com a independência de cada um é apresentada na próxima seção.

3.2.1. Usabilidade

Para sistemas de software convencionais, a Usabilidade vem freqüentemente em segundo plano, depois da funcionalidade, ou tem uma até mais baixa prioridade, no grau de importância para a realização do teste. Em aplicações de software voltadas para Web, a Usabilidade é um fator crítico e alguns autores chegam a definir como uma das áreas mais críticas em projetos Web. O sucesso de uma aplicação de Web depende em grande parte de um projeto intuitivo, uma estrutura de navegação fácil e compreensível, um formato(layout) claro e lógico, etc. Os usuários da Web não esperam ler manuais ou arquivos de ajuda (HOWER 1997)(MACINTOSH; STRIGEL, 2000).

Especialmente para aplicações comerciais na Web, a usabilidade é essencial: “A Web põe a experiência dos usuários de um Site em primeiro lugar, e o processo de compra e pagamento em segundo plano. E 90% dos Sites possuem projeto pobre em relação à Usabilidade (segundo a perspectiva dos usuários), assim que eles descobrem que o Site está carregado com gráficos e poucas informações úteis, eles vão a outro lugar. Se eles não podem achar o produto que eles querem, eles irão a outro lugar. E eles são hábeis para procurar os Sites em que eles sabem trabalhar” (NIELSEN; NORMAN 2000).

3.2.2. Projeto Gráfico

Outra questão, bem próxima da Usabilidade, é a atividade do projeto gráfico de uma Aplicação Web. Enquanto o Teste de Usabilidade focaliza principalmente na

facilidade de uso e preocupação com a aparência da aplicação, o Teste de Projeto tenta também examinar aspectos estéticos e a aparência geral da aplicação (RYDÉN; SVENSSON, 2001). A forma com uma aplicação Web é apresentada "tem um impacto considerável com usuários experientes. Uma página tem que parecer boa e a aparência deve ser consistente ao longo das páginas da aplicação" (KAUFMAN, 1999). Além disso, questão de compatibilidade (ver em 3.4.1.7) é outro aspecto a considerar, por exemplo, a resolução de tela e as características do navegador (browser).

Os fundamentos para o projeto gráfico de uma Aplicação Web devem ser registrados dentro um guia de estilo (SPLAINE; JASKIEL, 2001). Princípios de projetos gráficos para Aplicações Web e Web Sites podem ser encontrados em (LYNCHE e HORTON, 1999). Um guia com boas recomendações pode ser encontrado também em (SIEGEL, 1997).

3.2.3. Navegação e Ligações (Links)

Navegação descreve o modo como um usuário interage com a aplicação, através dos controles de interface do usuário e entre as páginas Web da aplicação (RYDÉN; SVENSSON, 2001). "A navegação de qualquer aplicação Web deve ser fácil, lógica, e intuitiva. Um usuário deveria poder navegar em todas as direções e não se perder ou não ser capaz de voltar a uma página anterior de forma indesejada" (KAUFMAN, 1999).

Em nível técnico, o teste de navegação inclui também a verificação de ligações entre as páginas, conhecidas como links. O objetivo dos links é conferir e assegurar que as ligações para páginas de Web internas e externas estejam funcionando. Especialmente as ligações para páginas Web externas precisam ser testadas freqüentemente, pois elas podem ficar indisponíveis inesperadamente (HOWER, 1997). Outro aspecto de teste relevante é o teste da correta implementação de links através de quadros (frames), imagens, mapas, redirecionadores, etc.

3.2.4. Acessibilidade

Projeto gráfico, navegação e questões de usabilidade afetam todos os usuários de uma aplicação web igualmente. Para um certo grupo de usuários, com certas inaptidões, as aplicações web precisam de cuidados especiais. Nielsen (1996) lista quatro tipos de inaptidões: (1) inaptidões visuais, (2) inaptidões auditíveis, (3) inaptidões motoras, e (4) inaptidões cognitivas.

Por exemplo, testando a conveniência de uma aplicação web por pessoas visualmente prejudicadas envolve a codificação apropriada de informação, rótulos de links e imagens, etc., para navegadores (browsers) de texto-para-fala especiais que lêem em voz alta as páginas. Embora esta possa ser uma restrição severa para "criativo" projeto gráfico de uma aplicação web, outras, menos mudanças significantes, também é sensato, por exemplo, escolher fonte e cores de imagem que se destinam a usuários daltônicos.

Além de muitos livros em Usabilidade, por exemplo, Nielsen (2000), o Site da Iniciativa Web Acessibilidade é uma fonte profunda de informação neste tópico.

3.2.5. Funcionalidade

O teste de funcionalidade de aplicações Web é bastante semelhante com o teste funcional de software convencional. Para definir a extensão do teste de funcionalidades, Nguyen (2001) lista algumas questões, as quais são tipicamente

levantadas ou apontadas quando da realização de teste de funcionalidade de aplicações *web*:

- Cada entrada e controle de navegação trabalha como o esperado?
- A aplicação realiza algo como o esperado?
- O que acontece quanto ao uso extremo especificado?
- O que acontece quando uma condição de erro ocorre?
- O que a experiência indica sobre as áreas mais problemáticas na aplicação?
- Porque o software falha? Como tirar lições aprendidas em uma série de ataques para expor o software a fracassos?

Até considerando documentos simples centrados em *Web Sites* já podem prover uma certa quantia de funcionalidade. Em particular, a interação com usuários têm que ser testada e as várias técnicas se concentram em, por exemplo, formulários, validação de dados, linguagens de *scripting* do lado cliente, HTML dinâmico, SSI, etc. (SPLAINE; JASKIEL, 2001) e em (RYDÉN; SVENSSON, 2001) *links*, formularios, *cookies*, índices *web*, linguagens de programação, componentes de interface dinâmicos, e bancos de dados como objetos de teste. Kaufman (1999) adiciona ainda segurança (como parte da aplicação *web*), controle de páginas *web*, armazenamento temporário de navegadores, frames, animações, áudio e vídeo, e impressão.

3.2.6. Qualidade de Código

Programas e Scripts compõem o comportamento dinâmico de um Web Site ou de aplicação Web. Um cuidado grande tem que ser levado em conta para assegurar que estes programas e scritps façam na verdade o que eles foram desenvolvidos para fazer. O nível do código fonte mede a garantia de qualidade do código, por exemplo, padrões de codificação e de documentação - tentando evitar erros. Ao lado de programas e scripts, há outras entidades satisfatórias para “testar o nível de codificação”: o código HTML em geral de páginas de Web, isto é, a conformidade com um certo padrão de HTML; imagens e o tamanho delas (largura e altura), tamanho (em bytes), e cor, profundidade, fontes, por exemplo, embutindo substituição e codificação de caractere; folhas de estilo; e tabelas. (SPLAINE; JASKIEL, 2001).

O Teste de entidades simples e únicas também é chamado de “teste de unidade” (POWELL, 1998): “Teste de unidade focaliza cada componente, página ou seção da aplicação separadamente...”. Testar a qualidade do código é tipicamente concluído em uma fase inicial do projeto, uma vez que uma simples unidade foi implementada.

3.2.7. Compatibilidade

O Teste de Compatibilidade preocupa-se principalmente com a compatibilidade das páginas web com os navegadores dos clientes. O número de navegadores diferentes e disponíveis no mercado é enorme e, além disso, existem múltiplas versões para cada navegador. Tipicamente todas as versões dos navegadores diferem em alguns aspectos, como por exemplo, no modo como eles apresentam uma página de web ou no modo que eles processam scripts no cliente. Além disso, navegadores provêem muitas características diferentes e opções de configuração, conduzindo até uma variação maior no comportamento (KAUFMAN, 1999) (POWELL, 1998).

Powell (1998) resume que o teste de compatibilidade com o navegador é basicamente: “Teste do navegador não só deveria cercar páginas carregadas corretamente e elementos trabalhando interativamente, mas assegurar que aqueles casos extremos sejam controlados. Aos usuários deveria ser permitido mudar configurações dos navegadores como tamanho de fonte, configurações de segurança, carga de imagem, cores, e qualquer outra preferência geralmente fixa para verificar se está conforme o uso. O teste com o navegador também deveria incluir uso básico do navegador, como impressão e redimensionamento, e apoio às características comuns como JavaScript, Java e CSS mesmo estando desabilitado.”

É quase impossível testar uma aplicação web ou web sites com todos os navegadores disponíveis no mercado, versões de navegadores e configurações. O que deve ser testado são as muitas combinações desejáveis de navegadores e suas características. Uma forma de encontrar as possíveis combinações com os navegadores é através da análise dos arquivos de Logs nos servidores Web para prever a audiência do Web Site (SPLAINE; JASKIEL, 2001).

Junto com o teste de compatibilidade, outros aspectos para o lado cliente têm que ser considerados também. Estes incluem sistemas operacionais, software adicional (como plug-ins), assim como, questões de hardware, como velocidade de processamento, resolução de tela, conexão com a Internet e o processamento máximo por unidade de tempo (throughput) (POWELL, 1998) (KAUFMAN, 1999) (GERRARD, 2000).

3.2.8. Banco de Dados

Aplicações Web e Web Sites incluem, tipicamente, uma grande quantidade de dados armazenados em banco de dados separados. Esses bancos de dados são geralmente usados por aplicações convencionais cliente/servidor na própria empresa. Além disso, aplicações web não só podem acessar esses banco de dados, mas também interagir com o servidor de sistemas legados e outros servidores de aplicação. É comum para aplicações de negócios (e-business) interagir com sistemas de outras empresas parceiras, como sistemas de compras ou serviços de cartões de créditos (SPLAINE; JASKIEL, 2001) (RYDÉN; SVENSSON, 2001).

O Teste de integração de todos os componentes e suas interações (inclusive software middleware, como drivers de banco de dados ODBC e JDBC), pode ser uma tarefa não muito fácil, pois são componentes de terceiros e nem sempre o código fonte está disponível. Testes de integração com sistemas legados cobrem acessos diretos e níveis de segurança, como funções de mapeamento de conversão de tipo de dados, sincronização e transferência de dados. A lista de teste com banco de dados apresentada em Splaine e Jaskiel (2001) menciona também que sistemas legados que estejam trabalhando “off-line” não devem ter um efeito negativo em uma aplicação Web e vice-versa.

3.2.9. Conteúdo

Semelhante à parte de impressão, o conteúdo tem que ser testado antes que uma aplicação Web ou Web Site seja disponibilizada ao público. Para o texto, ortografia e gramática tem que ser conferidas. Também, “páginas Web precisam ser inspecionadas com texto que esteja incompleto, não faz sentido, ou pareça fora do comum visualizar texto com falta de letra” (DUSTIN, 2001). Gráficos e imagens têm que ser conferidos

para ter certeza que a resolução e as cores são exibidos corretamente. O mesmo se aplica aos outros conteúdos de multimídia (POWELL, 1998).

Outro aspecto é em relação ao teste da precisão e integridade dos dados apresentados, como por exemplo, preços e medidas. Isto é importante, pois dados falsos podem conduzir facilmente a problemas legais (RYDÉN; SVENSSON, 2001) (ver mais sobre aspectos legais na próxima seção 3.2.10.).

Um problema organizacional relatado é a responsabilidade e a propriedade dos dados, isto é, quem provê os dados, quem confere e mantém os dados, e quem decide o que está disponível. Gerenciamento de versões é outro aspecto similar ao desta seção.

3.2.10. Aspectos Legais

Teste de aspectos legais e verificação de conteúdo estão sobrepostos. Como mencionado acima, o dado apresentado tem que ser preciso, seguro, e legal. “Em todo caso, o provedor dos conteúdos tem que ser, principalmente, confiável: Todo provedor de conteúdo tem a obrigação de ter certeza que o conteúdo não é ilegal” (COLLINS, 2002). Isto também inclui conferir que o conteúdo, isto é, textos, imagens, arquivos de áudio e de vídeos, não viole nenhuma restrição de direito autorais (POWELL, 1998).

Além disso, é sensato incluir condições de uso ou uma referência para o Web Site o qual se refere ao conteúdo, especialmente quando prover serviços via aplicação de Web. Quando a aplicação envolve transações financeiras, também deve incluir condições e termos gerais. Ao processar dados personalizados, a conformidade com regulamentos de proteção de dados tem que ser assegurada. Outro assunto controverso para se preocupar é a inscrição de nomes de domínio.

3.2.11. Internacionalização e Localização

Os termos internacionalização e localização são usados de forma trocada, dependendo do ponto de vista da organização, isto é, se a organização é orientada localmente ou globalmente. O significado principal em qualquer caso é o apoio de diferentes idiomas. “Como negócios tendem a continuar globalizados, a necessidade para localizar Web Sites fica mais difícil. Se a aplicação Web só pode ser vista em inglês, o negócio estará restrito as necessidades de só uma porcentagem pequena da população de Internet do mundo” (SPLAINE; JASKIEL, 2001).

Além das diferenças de idioma, há também outras armadilhas potenciais quando aplicações Web e Web Sites são desenvolvidoa para usuários do mundo inteiro, como formatos de data de sistemas de moeda e de medidas (GERRARD, 2000). Do ponto de vista técnico, atributos de idiomas tanto do lado servidor quanto do lado cliente, como codificação de caractere, têm que ser considerados (SPLAINE; JASKIE, 2001).

3.2.12. Desempenho, Carga e Stress

Teste de desempenho envolve verificar o tempo de resposta de uma aplicação de software. O tempo necessário para completar uma ação é usualmente comparado contra o tempo de uma ação executada por uma versão anterior do mesmo programa ou contra o tempo de uma ação idêntica em um programa semelhante. O tempo de comparação para uma parte do software também pode ser definido explicitamente pelo cliente no documento de especificação de requisitos. Ao documentar os requisitos, devem ser declarados os requisitos de desempenho em termos de números reais. (MACINTOSH; STRIGEL, 2000).

Os projetos de aplicações Web freqüentemente possuem definições e declaração de requisitos como “rápidas” e “aceitáveis”. Uma regra para o requisito de tempo de resposta utilizado em Nielsen (2000) pode ser adotada: menos de 0.1 segundo é o limite para o usuário sentir que o sistema está reagindo instantaneamente. Menos de 1.0 segundo é o tempo para o fluxo de pensamento de o usuário ficar interrompido e finalmente menos de 10 segundos é a quantia de tempo máxima que uma transação pode transcorrer enquanto mantém a atenção do usuário com a aplicação. (SPLAINE; JASKIEL 2001).

Existem múltiplos fatores que influenciam o desempenho global de uma aplicação Web. Em geral máquina dos clientes, máquinas servidoras, servidores Web, servidores de banco de dados e a rede devem ser medidas. Para estimar a influência de diferentes fatores e descobrir onde os pontos críticos de transmissão de dados, os testes devem ser integrados com a implementação da aplicação Web (DUSTIN, 2001). Somente um número limitado desses fatores pode ser controlado, mas o impacto desses fatores juntos é relevante. Outros fatores não técnicos têm que ser levado em conta também, quando existe um planejamento de teste de carga de uma aplicação web: tempo do dia, publicidade e mudanças (HOWER, 1997).

O que faz as medidas de desempenho serem complicadas é o fato que os diversos fatores dependem do número de usuários acessando simultaneamente a aplicação Web ou o Web Site. Teste de carga é usado para descobrir, como o sistema reage a uma certa carga, especificada como uma certa quantia de usuários, executando uma certa quantia de transações e transferindo uma certa quantia de dados. A carga aplicada é a carga esperada que o sistema receberá em um ambiente do mundo real sobre condições realísticas (MILLER, 2001).

Por outro lado, o Teste de Stress tenta “derrubar” a aplicação Web ou o Web Site, expondo a uma carga máxima sob um longo período de tempo. O propósito de um teste de Stress é determinar onde estão os limites do sistema e descobrir o que acontece ao sistema quando existem operações em cima desses limites, isto é, se o sistema deixa de funcionar (KALLEPALLI, 2001-b).

Splaine e Jaskiel (2001) sugerem a realização de um terceiro tipo, chamado de Teste de Smoke, em adição ao teste de Carga e teste de Stress, que significa avaliar se uma atualização de software está realmente pronta para ser testada. O Teste spike/bounce (picos e saltos) tem como objetivo descobrir como o sistema suporta picos de cargas e mudanças bruscas de carga. Outras áreas relacionadas com testes de desempenho são comentadas nas próximas seções como Escalabilidade, Disponibilidade, Confiança e Recuperabilidade.

3.2.13. Escalabilidade

Teste de Escalabilidade direciona alguns problemas bem parecidos com a realização do Teste de Desempenho. Splaine e Jaskiel (2001) entendem o teste de escalabilidade como a habilidade que um sistema possui para suportar um aumento significativo na sua capacidade para acomodar crescimento no futuro.

3.2.14. Disponibilidade e Confiabilidade

O Teste de Disponibilidade é medido em termos de “uptime” que é a proporção entre o tempo que um serviço Web está disponível para uso e o tempo total do período de medição (GERRAD, 2000). Confiabilidade é geralmente medido pelo tempo médio

entre as falhas, ou o número de falhas que duram para um certo intervalo durante um período especificado de tempo (SPLAINE; JASKIEL, 2001).

Normalmente os dois tipos de teste revertem para o Teste de Carga para monitorar a aplicação sob teste por um período de tempo estendido. O propósito dos testes é descobrir fracassos que evoluem lentamente com a evolução do tempo. Em Splaine e Jaskiel (2001) esta abordagem é chamada de Teste de Resistência. Outras possibilidades de abordagens mencionadas são Teste de Baixo Recurso, Teste de Volume e Picos de Cargas.

3.2.15. Segurança

Segurança pode ser também considerada como parte da funcionalidade de uma aplicação *Web* ou de um *Web Site*, isto é usualmente o caso quando a segurança é um requisito explícito ou pode ir longe além da aplicação e estendida para segurança no servidor, segurança na rede, e outros tópicos (POWELL, 1998).

Nguyen (2001) cita Java, ActiveX, *cookies*, vírus, ataques de serviços negados, e ataques físicos como possíveis locais de vulnerabilidade. Dustin (2001) agrupa vulnerabilidade e riscos em:

- Questões de Segurança na Internet - originadas por razões históricas, a Internet e seus respectivos serviços originais são inerentemente inseguros;

- Segurança em Serviços e Sistemas Operacionais – *Web Sites* ou até mesmas aplicações *Web* são arquiteturas complexas de multi-camadas que contém freqüentemente problemas (*bugs)* que podem ser explorados; e

- Riscos de Segurança *OutSourcing* – exemplos são os provedores de serviços de internet para aplicações *Web* e *Web Sites*, contendo dados proprietários.

A respeito da segurança, os erros em produtos de terceiros podem ser uma porta para os problemas de segurança que comprometem o *Web site* inteiro ou aplicação. MacIntosh e Strigel (2000) destacam que estes erros são difíceis de localizar desde que na maioria dos casos, a execução dos componentes de terceiros não está disponível e o teste daqueles componentes tem que ser restringido a teste de caixa preta. Em geral, duas maneiras para executar o teste de segurança são sugeridas: auditoria de segurança e a intrusão, isto é, a simulação de invasão (GERRARD, 2000).

3.3. Classificação de Aspectos de Qualidade

Ramler (2002) organizou e classificou os aspectos de qualidade para o teste de aplicações Web em relação a três dimensões: qualidade, características e fase. Essa combinação resulta um esquema que pode ser usado para definir os diferentes passos que devem ser realizados no processo de teste de aplicações Web. A organização dos aspectos de qualidade de acordo com a figura abaixo depende principalmente dos requisitos de um certo sistema. A Figura 3.3 apresenta o esquema da classificação.

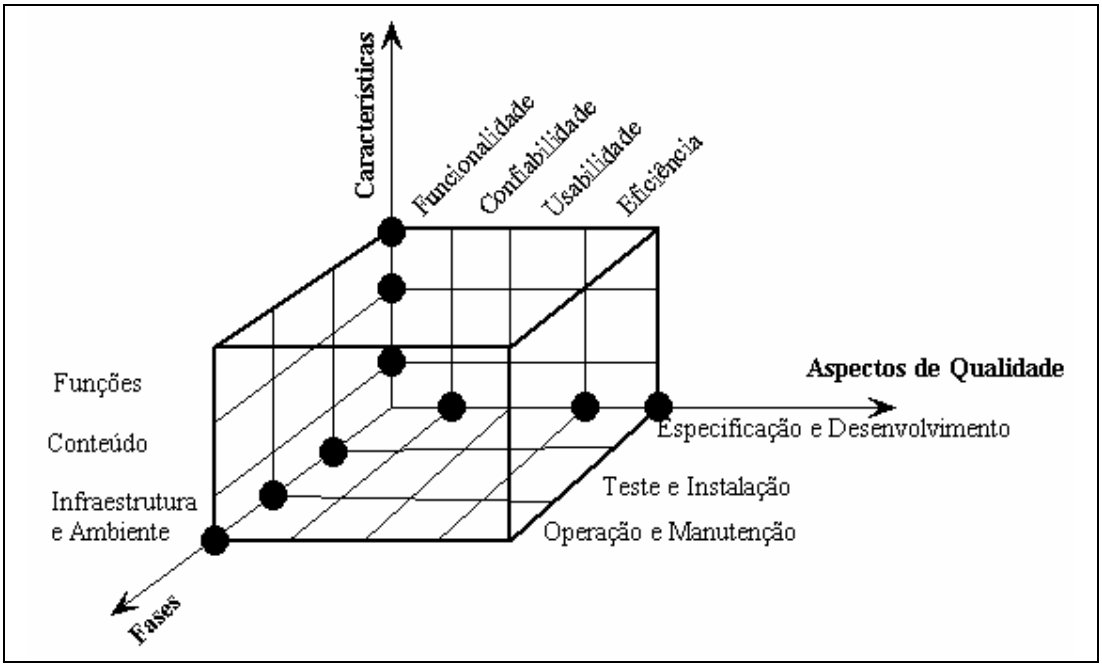


FIGURA 3.3 - Esquema de Testes de Aspectos de Qualidade

A dimensão de qualidade descreve os aspectos de qualidade que são subjetivos para os testes, e vai depender de quem tem interesse na aplicação, por exemplo, clientes, usuários, desenvolvedores, gerentes e o público em geral. A dimensão de aspectos de qualidade tem origens nos aspectos de garantia de qualidade, os quais provêem um extensivo conjunto de características de qualidades que podem ser usados para uma classificação mais genérica. Estas características de qualidade são partes de modelos de qualidade apresentadas em Boehm (1996) e McCall (1977), implementadas como métricas para medir a qualidade de certos aspectos de um sistema. Olsina (2001-b) apresenta uma lista de requisitos de características de qualidade, que descreve o que deve ser levado em consideração para aplicações específicas, como por exemplo, Web Sites acadêmicos.

A dimensão de características, a qual a aplicação contém, é usualmente definida na especificação de requisitos como requisitos funcionais, isto é, uma característica é um conjunto de requisitos funcionais relatados logicamente que provêem uma capacidade para o usuário. Testes de características é um dos principais conceitos de teste de software convencional. Essa dimensão pode ser classificada em: Funções, as quais descrevem a soma de que o sistema faz, todos aqueles aspectos que contribuem para o comportamento do sistema; Conteúdos incluem os dados que são partes do sistema ou desenvolvida como parte do sistema, assim como a estrutura e apresentação. O conteúdo, como usado neste contexto, deverá primariamente ser definido como parte de um requisito funcional; Infraestrutura e Ambiente, cobre todos os conceitos ligados aos requisitos do sistema em relação à largura de banda de transmissão, configuração de servidores, localização geográfica de servidores, etc. A razão de considerar a infraestrutura e o ambiente são bem simples: no final, aspectos de qualidade são afetados por qualquer parte do sistema e não fará diferença para o usuário porque um sistema falhou.

A última dimensão na classificação de Ramler (2002) é a dimensão de Fase. A preocupação de teste dos aspectos de qualidade e características em diferentes fases abrange um ciclo de vida inteiro de um sistema. Embora todo projeto tenha uma forma

específica e tarefas que necessitam de um fluxo de trabalho em especial, uma estrutura global usualmente segue um típico processo de engenharia dividido em definição de requisitos, projeto, desenvolvimento, integração instalação, operação e manutenção (SOMMERVILLE, 2001). Desta forma, a abordagem utilizada tem como objetivo organizar as atividades de teste de acordo como os níveis de teste, ou seja, teste de unidade, teste de integração e teste de sistema.

O esquema da classificação apresentado permite visualizar as atividades de testes de aplicações Web de uma forma que todos os aspectos estejam relacionados. O objetivo é um melhor entendimento da área de testes de aplicações Web como forma de prover procedimentos adequados para a garantia de qualidade.

3.4. Estrutura do Teste de Aplicações Web

Agora, que uma lista dos tipos e objetos de teste foram apresentados e classificados, explicando o que testar, a pergunta é, como esses testes podem ser estruturados para formar uma aproximação significante para testar aplicações Web. A situação ideal para testadores seria ter um guia para dividir os principais objetos de teste dentro de objetos menores e finalmente mais adiante declarações ou questões de testes, que podem ser verificadas e marcadas, isto é, respondendo sim ou não. O resultado seria uma hierarquia estruturada em árvore de teste, onde todos os testes de níveis mais baixos formem juntos os testes para o nível acima na árvore e todos os resultados desses testes acumulariam valores para o resultado dos testes finais.

Ainda segundo (Ramler, 2002), a estruturação dos testes não é uma linha reta que deve ser seguida. Os testes listados na Secção 3.2 são entrelaçados, possuem muitas dependências, e algumas vezes se sobrepõem. Alguns testes focalizam os mesmos objetos de teste, mas fazem isso de pontos de vista diferentes; outros testam objetos diferentes, mas focalizam um problema semelhante; e outros, novamente, concentram-se em um certo subconjunto de outras questões de teste. O que é verdade para fatores de qualidade em aplicações parece solicitar outros tipos de teste de um modo semelhante: a interdependência pode ser complementar, conflitante, ou indiferente. A Figura 3.4 tenta ilustrar a interdependência complexa dos tipos de teste diferentes: as linhas ligando cada tipo de teste são possíveis dependências; as áreas cinzas sugerem uma relação semântica íntima.

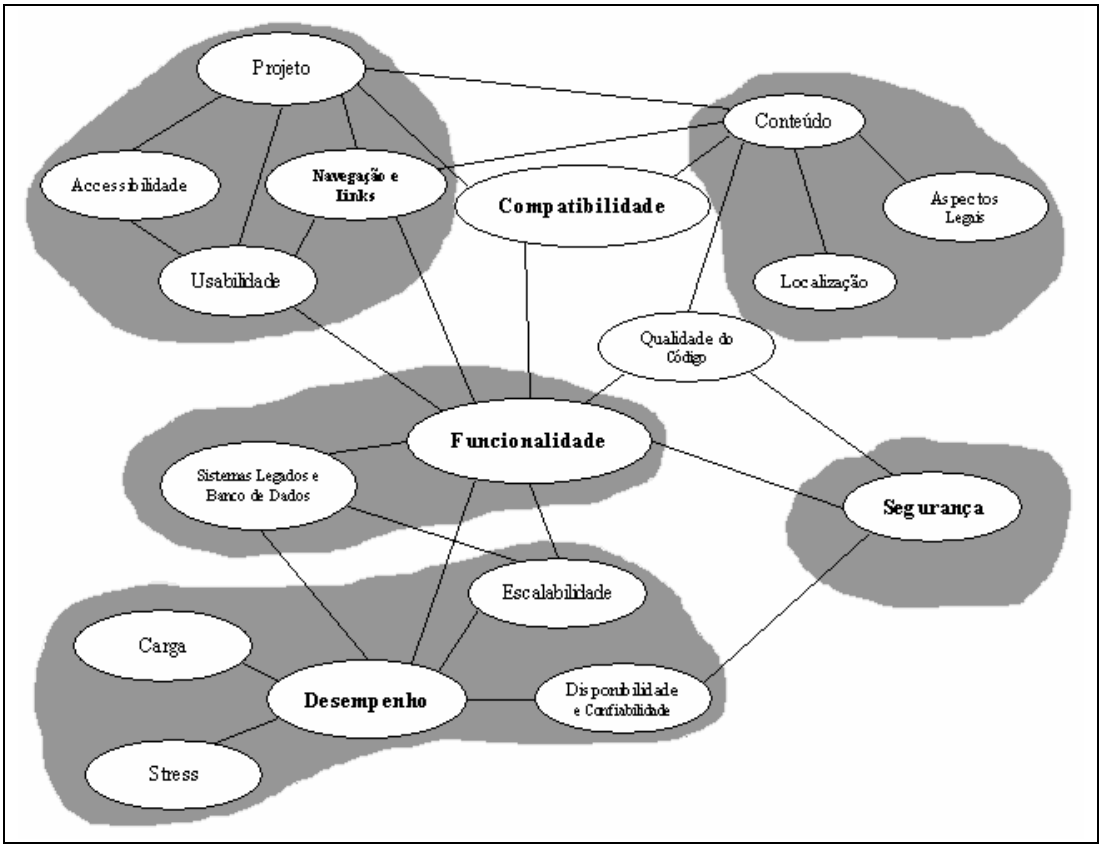


FIGURA 3.4 - Estrutura de Teste de Aplicações Web

3.5. Ferramentas de Teste de Aplicações Web

A realização do teste de software manualmente é tediosa e extremamente propensa a erros nos resultados, especialmente, quando testes têm que ser repetidos, como, por exemplo, testes de regressão. Ferramentas automatizadas permitem aumentar a eficiência do teste. Outros tipos de testes, como testes de carga que simulam centenas e milhares de usuários ou testam a resistência de aplicações por um período de tempo estendido, não podem ser realizados de forma manual e de modo econômico sem o apoio de ferramentas automatizadas.

Esta Seção trata das ferramentas de apoio ao teste de aplicações Web. É apresentada primeiramente uma motivação para o uso das ferramentas, seus benefícios e também discutidas algumas desvantagens. São realizados comentários também sobre quais as ferramentas para a realização de testes de aplicações Web e como essas ferramentas podem ser categorizadas. Uma avaliação é realizada com essas ferramentas e através de uma comparação são mostradas algumas características das principais ferramentas.

3.5.1. Benefícios e Problemas na Automação de Testes

A automatização do teste se faz através de um software que automatiza qualquer aspecto de teste de uma aplicação de software. Isso inclui a capacidade para gerar entradas de teste e resultados esperados, executar pacotes de teste sem a intervenção

manual e avaliar os resultados se a aplicação passou ou não no teste. Para ser efetivo e repetido, o teste deverá ser automatizado (BINDER, 1999).

Ferramentas automatizadas de teste ajudam em muito a realização do teste. Porém, as aplicações de ferramentas não conduzirão necessariamente para a automação dos testes. Por exemplo, ferramentas que executam testes de cobertura somente em um aspecto de automatização dos testes. Além disso, enquanto a automação do teste possui benefícios significativos, existem desvantagens em potenciais que devem ser lembradas. Rydén e Svensson (2001) e Fewster (1999) apresentam os seguintes benefícios e problemas de automatização de teste:

Benefícios

- A automação dos testes permite executar os mesmos testes em versões futuras da mesma aplicação;
- Automação de testes resulta na execução mais freqüentemente de testes;
- Testes automatizados são executados de forma consistente, exata e da mesma maneira todo o tempo;
- Testes automatizados permitem verificação rápida e eficiente de problemas corrigidos através do teste de regressão;
- Testes automatizados podem ser utilizados para suportar a depuração de programas;
- Testes automatizados podem melhorar a produtividade;
- A automatização de tarefas complicadas permite mais tempo gasto no projeto de casos de testes, para alcançar uma melhor taxa de cobertura.
- A automação de teste é necessária para testes longos e complexos, como teste de carga e de resistência;
- Testes automatizados evitam erros humanos, introduzidos quando da entrada manual de dados e a comparação com a saída.

Desvantagens

- Uma boa organização de testes estruturados é necessária antes da automação;
- Para criar e executar testes mais específicos, as ferramentas precisam ser entendidas completamente;
- Quando um teste é realizado, muitos dos erros já foram encontrados;
- Pode ser difícil distinguir onde as falhas ocorreram se na ferramenta ou na aplicação;
- A criação de scripts de testes automáticos, por um lado apresenta uma baixa manutenção;
- Testes automatizados são mais caros do que testes manuais, quando só ocorrem uma vez ou algumas vezes.

Com a lista de vantagens e desvantagens mostrada acima, é importante também notar o estágio de teste para verificar se a automação faz sentido ou não. O tempo certo para a automação do teste de acordo com Wu (2001) não acontece até que a estrutura fundamental de um sistema esteja estável. A partir deste ponto podem ser implementados testes sem o risco de ficar fora do prazo no esforço de requisitos e manutenção significativos.

Linz e Daigl (1998) mediram as despesas para um teste de Interface Gráfica com o Usuário (GUI) automatizado e os comparou às despesas de realizar os mesmos testes de forma manual, com o objetivo de achar qual o número de repetições de teste no qual os testes automatizados são executados com menos custo que os testes manuais. Em média, os custos para testes automatizados são entre 125% e 150% de custos de testes manuais, quando só corre uma vez. Assim, depois da segunda ocorrência, testes automatizados já podem ser considerados com um custo mais eficiente.

3.5.2. Categorias e Ferramentas de Teste

Um vasto número de ferramenta está disponível para o teste de software convencional. Hower (2001) listas não menos que 100 ferramentas de teste diferentes e técnicas como ferramentas de análise de fluxo, teste de cobertura, simuladores ambientes e ferramentas medidas de complexidade.

Binder (1999) identifica oito tipos principais de ferramenta de teste diferentes:

1. Utilitários de dados, inclusive geradores de dados de teste, editores, formatadores e analisadores;
2. Simuladores, os quais configuram o ambiente e geram entradas de maneira controlada e de forma repetitiva;
3. Comparadores, para arquivos e fluxo de dados;
4. Ferramentas de *Capture/Replay,* incluindo características de *scripts* e testes de regressão;
5. Analisadores de cobertura e de rastreamento;
6. Analisadores de programa, isto é, descobridores de caminhos e ferramentas de reestruturação;
7. Depuradores interativos;
8. Ferramentas CASE para estender os testes, podem ser usadas para produzir planos de testes a partir de especificações e prover conjunto de testes de forma estruturada.

Outras formas para categorizar ferramentas de teste enfatizam: o aspecto estático ou dinâmico do teste (POMBERGER; BLASCHEK, 1996), a origem dos dados de onde podem ser derivados casos de teste (baseados em código, baseada em interface e baseado em especificação) (RYDÉN; SVENSSON, 2001), os tipos de testes suportados (DUSTIN, 2001), ou outras possibilidades no uso de ferramenta (RYDÉN; SVENSSON, 2001).

Esta seção trata de ferramentas de teste voltadas para aplicações Web. Muitas das ferramentas de teste usadas para software convencional são também utilizadas para aplicações Web, especialmente aquelas que cobrem tarefas básicas como editores e analisadores. Ainda, ferramentas adicionais são necessárias para o teste de propriedades específicas de Web Sites ou aplicações Web, como hiperlinks, por exemplo.

Hower (2001) estruturou e apresenta referências para ferramentas de teste nas seguintes categorias: ferramentas de teste de carga e desempenho, ferramentas de teste Java, verificadores de links, validadores de HTML, validadores HMTL e verificadores de links de uso livre, para validação e verificação de programas em Perl e C, ferramentas de teste funcional e de regressão, ferramentas de teste de segurança de Web Sites, serviços que monitoram Web Sites externos, ferramentas de gerenciamento e

administração de aplicações Web, ferramentas de análise de transações (Logs), e outras ferramentas.

A seguinte lista de tipos de ferramentas para aplicações Web foi extraída de (DUSTIN, 2001) e (NGUYEN, 2001). Os autores apresentam ferramentas e fornecedores para todos os seguintes tipos de ferramentas:

3.5.2.1 Analisadores Estáticos

A idéia básica de analisadores estáticos é verificar a conformidade do código de fonte com as regras de codificação específicas e padrões. Segundo Nguyen (2001), "estático" significa que o código fonte é que é analisado. Os resultados de uma análise estática são relatórios analíticos e de erro.

Exemplos para tais ferramentas são: Validadores HTML, usados para verificar a conformidade com certos padrões HTML, Analisadores de Código para linguagens de programação, Otimizadores de Imagens, o qual podem ser usados para verificar o tamanho e estimar o tempo de "download" para as imagens, e Verificadores de Gramática, usado para conferir erros nos textos.

3.5.2.2 Detectores de Erros Dinâmicos

Ao contrário do item anterior, detectores de erros dinâmicos exigem que o código seja compilado e executado, com a ferramenta sendo usada para achar e erros em tempo de execução. Conseqüentemente, tais ferramentas são chamadas dinâmicas. Um típico exemplo para detectores de erros dinâmicos são ferramentas que verificam problemas de armazenamento e verificadores de Links, usados para descobrir Links quebrados.

3.5.2.3 Ferramentas de Teste de Carga e de Desempenho

Não é um tipo de ferramenta utilizada somente para aplicações Web. Mas muito importante na medida de desempenho. As ferramentas nessa categoria são usadas para avaliação e testes de desempenho, de carga, estresse, escalabilidade e de confiabilidade.

O conceito básico de como as ferramentas de teste de desempenho trabalham é: (1) enviar uma requisição simulada do usuário para o servidor, (2) medir o tempo de resposta e (3) avaliar o resultado. O foco de tais ferramentas é que um grande número de usuários, fazendo acesso simultaneamente, a uma aplicação Web pode ser simulado. Além disso, testes podem ser executados por um longo período de tempo para testar a resistência.

Ferramentas de teste de carga e de desempenho são relacionadas para desempenhar monitoramento de teste de implementação de requisição, assim como ferramentas Capture/Replay, as quais são geralmente usadas para armazenar requisições.

3.5.2.4 Ferramentas Capture e Replay

Ferramentas Capture/Replay são aplicadas para teste de interface com usuários, teste de funcionalidade e teste de regressão. Essas ferramentas são capazes de capturar requisições do usuário e fazer uma nova execução das requisições sempre da mesma forma, com pouca ou nenhuma interação humana.

Essas ferramentas têm a capacidade de reconhecer controles da interface gráfica com o usuário, tais como botões, tabelas, links, Applets Java, em páginas Web. Durante a fase de captura, essas ferramentas localizam eventos de entrada (geralmente através de teclado e mouse) e como eles são aplicados para controles específicos de objetos das GUI. Os eventos representam atividades do usuário e são convertidos em scripts, que mais tarde, a máquina de playback usará como entrada para a nova execução das atividades armazenadas. Durante a execução a informação do estado do programa, assim com as saídas dos resultados são comparadas com os resultados originais. Se existe qualquer discrepância, a ferramenta indica a condição (Nguyen, 2001).

Splaine e Jaskiel (2001) apresentam ainda três categorias para esse tipo de ferramenta:

- Capture/playback – são capazes de capturar interações dos usuários e então executar novamente como se fosse um novo usuário.
- Variable Capture/playback – também são capazes de capturar interações com o usuário e permitem que modificações no script sejam realizadas antes da nova execução.
- Variable Capture/Variable playback – além de capturar interações com usuário e permitir modificações na entrada dos testes, neste caso é permitido também a modificação da saída para ser comparada na execução.

Quanto mais flexível a ferramenta, maior o custo e de difícil uso. Diferentes técnicas são usadas para capturar páginas web. Essas ferramentas seguem o Modelo de Objetos de Documentos (DOM) utilizados pelos principais fabricantes de navegadores.

3.5.2.5 Ferramentas de Teste de Segurança

Estas ferramentas podem ser usadas para testar a segurança global de uma aplicação Web, como a infra-estrutura, verificando os riscos de segurança, normalmente, conhecido como vulnerabilidade.

Dustin (2001) relata que, infelizmente, as maiorias das ferramentas disponíveis são orientadas para tarefas pequenas, como leituras de portas de comunicação, ou verificação do sistema operacional e serviços de configuração de um servidor para vulnerabilidades conhecidas. Verificação de segurança avançada envolve a investigação das interfaces da aplicação e componentes de Web Site específicos de modo que não podem ser controlados automaticamente por tais ferramentas. Outros exemplos típicos de ferramentas para testar a segurança são monitoramento da rede, ou verificadores de vulnerabilidade de sistemas.Ferramentas Capture/Replay são aplicadas para teste de interface com usuários, teste de funcionalidade e teste de regressão. Essas ferramentas são capazes de capturar requisições do usuário e fazer uma nova execução das requisições sempre da mesma forma, com pouca ou nenhuma interação humana.

3.5.2.6 Ferramentas de Teste de Compatibilidade

A verificação de compatibilidade com diferentes tipos de Navegadores, Sistemas Operacionais e suas respectivas versões são tarefas manuais. Porém, existem algumas ferramentas e serviços disponíveis para suportar o teste de compatibilidade. Um exemplo pode ser espelhamento da aplicação, o qual permite instalar sistemas operacionais diferentes e diferentes navegadores, isto é, diferentes versões de um navegador, no mesmo computador.

3.5.2.7 Ferramentas de Teste de Usabilidade

Outro tipo de teste que ainda se justifica na confiança de um testador humano é o teste de Usabilidade. Novamente, existem algumas poucas ferramentas que apóiam este tipo de teste, por exemplo, verificar de forma estática se existe uma complacência com padrões e recomendações de projetos Web.

3.5.2.8 Ferramentas de Planejamento de Teste

Ferramentas de apoio ao planejamento de teste são utilizadas para dar suporte a equipe de teste com o objetivo de um melhor gerenciamento da atividade de teste. Existem ferramentas que controlam os requisitos, casos de testes, execução dos testes e apresentam alguns relatórios de saída.

3.5.2.9 Outros tipos de Ferramentas úteis

Muitas outras ferramentas, algumas que até mesmo não têm a ver com teste diretamente, podem ser úteis também para o teste de aplicações Web. Exemplos são ferramentas de análise de perfis, analisadores de cobertura, ferramentas de reparo de defeitos, ferramentas de administração de casos de testes, e ferramentas de administração de versões.

3.5.3. Avaliação das Ferramentas

Essa seção apresenta os principais critérios de avaliação das ferramentas de teste de aplicações Web segundo Dustin (2001), Robison (2001) e Rydén e Svensson, (2001). O objetivo é apresentar os pontos principais nas avaliações apresentadas pelos respectivos autores. Essas avaliações foram realizadas com ferramentas de teste dos principais fabricantes: Compuware, Empirix, Mercury Interactive, Rational Software e Segue Software.

3.5.3.1 Avaliação por (Dustin, 2001)

Uma extensa lista de avaliação de ferramentas é apresenta em Dustin (2001). Porém, essa avaliação está mais voltada para característica por empresa do que pelas próprias ferramentas, pois os critérios não foram aplicados em cada ferramenta em separado, mas para cada critério avaliado de cada empresa é feito um critério em particular. Os 19 critérios diferentes de avaliação usados são categorizados como o seguinte:

- ✓ Características Gerais das Ferramentas de Teste
    - Facilidade de Uso;
    - Capacidade de Customização da Ferramenta;
    - Suporte a Plataforma;
    - Características de Teste das Linguagens;
    - Teste de Ferramentas de Banco de Dados.

- ✓ Capacidade de Execução de Teste

    - Características de controle dos Testes;
    - Execução de testes distribuídos;
    - Gerenciamento de teste.

- ✓ Capacidade de Integração

    - Integração da Ferramenta: incluem integração com as seguintes ferramentas: modelagem e projeto, teste de unidade, gerenciamento de teste, gerenciamento de requisitos, reparação de erros e gerenciamento de configuração.

- ✓ Capacidade de Geração de Relatórios de Teste

    - Relatórios de Resumo;
    - Apresentação de Relatórios de Teste;

- ✓ Capacidade de Análise e Teste de Desempenho

    - Características de Teste de Carga de estresse*:* geração de cenários automáticos de carga, simulação de usuários virtuais, etc;
    - Características de Teste de Monitoramento de Desempenho.

3.5.3.2 Avaliação por (Robison, 2001)

Robinson (2001) realizou uma avaliação com os mesmos fabricantes da seção anterior. Os critérios avaliados foram: características de Capture/Replay, características de teste de aplicações Web, funções de dados, mapeamento de objetos, teste de imagem, recuperação de testes/erros, suporte ao ambiente, integração, custo, facilidade de uso, extensibilidade de linguagens e suporte do fabricante.

3.5.3.2 Avaliação por (Rydén and Svensson, 2001)

Rydén e Svensson (2001) também incluem uma avaliação com dois fabricantes, Rational (com as ferramentas Rational Robot e TestManager) e Mercury (através das ferramentas Astra Quick Test e Astra Load Test). Os critérios de avaliação usados foram: possibilidades e funções de teste oferecida pela ferramenta, usabilidade, edição de scripts, análise de erros e outras funcionalidades como integração com outras ferramentas para testes avançados.

Rydén e Svensson (2001) procuram enfatizar que a escolha da ferramenta depende das seguintes questões: tamanho do projeto de teste, tamanho da aplicação, tecnologia utilizada na aplicação, estágio do processo de teste, recursos disponíveis e tipos de teste.

3.5.4. Resultados da Avaliação

Considerando os principais aspectos citados pelos autores na avaliação da ferramenta anteriormente, a Tabela 3.2 apresenta os resultados das avaliações realizadas com algumas ferramentas utilizadas no decorrer dos testes neste trabalho considerando as características para a utilização dessas ferramentas.

TABELA 3.2 - Resultado da Avaliação da Ferramenta

Características	e-Test	LoadTest	Alert*Link*	QuickTest
Capacidade de Integração	Sim	Sim	Sim	Sim
Relatórios de Resumo	Sim	Sim	Sim	Sim
Capacidade de Customização	Sim	Sim	Não	Não
Facilidade de Uso	Sim	Médio	Sim	Sim
Abrangência de Aplicações	Alto	Alto	Baixo	Baixo

A Tabela 3.3 apresenta algumas das principais ferramentas de teste para aplicações Web quanto aos principais e maiores fornecedores, os quais apresentam, geralmente, várias soluções na área de teste.

TABELA 3.3 - Ferramentas por Fornecedor

Fabricante	**Desempenho**	**Funcional/ Regressão**	**Planejamento e Gerenciamento de Aplicações**
Mercury	LoadTest	QuickTest	TestDirector
Rational	Site Load	Robot	SiteCheck
Compuware	QALoad	QARun	QADirector
Segue	SilkPerformer	SilkTest	SilkVision
Empirix	e-Load	e-Tester	E-Monitor

3.6. Conclusões

Este capítulo apresentou uma visão da área de teste de aplicações Web, dando ênfase aos diversos tipos de teste pesquisado na bibliografia. Notou-se o desafio do emprego de testes em aplicações baseada na Web em função da complexidade do ambiente e de características específicas de tais aplicações. Diversos tipos foram apresentados, cada um mostrando seus benefícios e dificuldades. Esses tipos de testes podem ser estruturados por categorias como forma de melhorar o entendimento das principais áreas de teste.

A utilização de ferramentas de teste é de grande importância, visto que, alguns testes são impraticáveis, para não afirmar, quase impossíveis. Existem diversas categorias de ferramentas de teste para aplicações Web, porém algumas áreas de teste não são realizadas de forma totalmente automatizada, como o teste de Compatibilidade. A utilização dessas ferramentas possui vantagens e desvantagens, é necessário identificar os objetivos de teste.

O teste de aplicações Web e Web Sites envolve muitas áreas e questões que até então, não se apresentavam em aplicações de software tradicional.

4. Processo de Teste de Aplicações Web

O processo de teste para software convencional pode se aplicado também para aplicações baseadas na Web, porém algumas características particularidades neste tipo de aplicação requerem alguma atenção especial. Após o levantamento dos tipos de testes no capítulo anterior, é importante definir que atividades devem ser seguidas no processo de teste. Essas atividades estão relacionadas também com as fases de teste que uma aplicação de software deve passar.

O objetivo de um processo de teste de aplicações Web é fornecer uma descrição clara e concisa do que deve ser testado e como os testes devem ser realizados. O processo tem que ser estruturado e administrado com a finalidade de manter uma seqüência lógica de atividades até que seja alcançado o objetivo final. Este capítulo aborda os conceitos aplicados às atividades e ao processo de teste de software, e como esse é estendido para aplicações baseadas na Web e para Web Sites. O processo apresentado é implementado na ferramenta proposta por este trabalho, no capítulo seguinte.

Este capítulo apresenta, na Seção 4.1, as atividades do processo de teste de software. A abordagem de utilização do processo de teste de aplicações Web é mostrada na Seção 4.2. Na Seção 4.3 são apresentadas algumas ferramentas de suporte ao processo de teste.

4.1. Processo de Teste

Em geral, as atividades de teste não mudam quando aplicadas a aplicações baseadas na Web. Os testes incluem as seguintes atividades principais: planejamento de teste, projeto e implementação de casos de teste, execução dos testes, avaliação do testes e gerenciamento do teste (Rydén; Svensson, 2001). A seguir é apresentada uma breve descrição dessas atividades.

4.1.1. Planejamento de Teste

De acordo com o padrão ANSI/IEEE 829-1983 para a documentação de teste de software, um plano de teste é definido como um documento que descreve o escopo, abordagem, recursos e escalonamento da realização das atividades de teste. O documento identifica itens de teste, as características a serem testadas, as tarefas de teste, quem realizará cada tarefa e qualquer risco requerendo plano de contingência (Kaner, 1999).

Segundo Nguyen (2001), o planejamento de teste para aplicações Web é semelhante ao planejamento do software tradicional, porém merece um cuidado especial em função de várias tecnologias empregadas e a quantidade de funcionalidades projetadas, assim o planejamento de teste é sempre importante para o gerenciamento dos testes aplicados somente em aplicações Web.

4.1.2. Projeto e Implementação de Caso de Teste

Casos de teste descrevem um único objetivo de teste especificando o que testar e como testar, inclusive condições prévias, condições implementadas e o ambiente de

teste. Bons casos de teste satisfazem os seguintes critérios segundo Kaner (1999), uma probabilidade razoável de descobrir um erro, não-redundância e não ser muito simples nem muito complexo. Rydén e Svensson (2001) enfatizam a importância de casos de teste sustentáveis e reutilizáveis.

Para o projeto de casos de teste, métodos como particionamento de equivalência, análise de limite-valor e outras técnicas são conhecidas. Em Nguyen (2001), são apresentados como esses métodos podem ser aplicados em testes de aplicações Web. Como a Engenharia para Web inclui uma ampla extensão de tarefas diferentes, métodos adicionais, como, por exemplo, originados da área de teste de Usabilidade (HOM, 1998), podem se tornar significativamente mais importantes também.

Rice (2001) apresenta uma abordagem de como desenvolver casos de teste e scripts de teste para aplicações Web. Primeiro é definido os componentes dos casos de teste com a seguir:

- ID: um número único que identifica um caso de teste. O identificador não implica em uma ordem seqüencial no momento de execução do teste;

- Condição: isto deve ser um evento que deverá produzir um resultado a ser observado;

- Resultado esperado: isto é um resultado de uma invocação de uma condição de teste.

- Procedimentos: é o processo que o testador necessita para realizar as condições de teste e observar os resultados. Um procedimento caso de teste deverá limitar os passos necessários para a realizar um único caso de teste.

- Aprovado/Falha: são os indicadores de saída dos casos de testes. Após a execução dos casos de teste, o resultado deve ser de sucesso ou falha.

- Número de defeito: se a identificação do defeito do caso de teste for registrada, este componente do caso de teste apresenta uma forma de ligar os casos de teste para um específico relatório de defeitos.

Algumas estratégias de projeto casos de teste também são apresentadas em Rice (2001). Em algumas vezes é difícil decidir quais os casos de testes deverá trabalhar melhor, mas existem algumas estratégias em comum. Assim é definida a necessidade de considerar as questões:

- O tipo de aplicação a ser testada;
- A natureza do público de usuários;
- A necessidade de repetir o teste;
- O nível de riscos;
- O grau de mudança.

4.1.3. Execução de Testes

Executar os testes é a tarefa operacional de definir os pré-requisitos, executar o caso de teste, e avaliar os resultados comparando os resultados atuais com os resultados

esperados. Todos os erros encontrados têm que ser documentados e informados. Enquanto os princípios básicos do teste convencional são os mesmos para teste de aplicações *Web*, o ambiente de teste é freqüentemente mais complexo devido à complexidade dos fatores envolvida execução de uma aplicação de *Web*.

4.1.4. Avaliação de Resultado

Essa atividade verifica se os resultados estão coerentes com os requisitos esperados. A avaliação procura também relacionar os diversos resultados de testes com o objetivo de prover informações que podem indicar propriedades que são inerentes a vários tipos de testes. Por exemplo, em aplicações *Web*, resultados de testes de navegação (*links*) estão relacionados ao testes funcionais, como foi apresentado na Figura 3.4. Desta forma, pode-se apontar que para um determinado *link* quebrado, um caso de teste não será executado de forma satisfatória.

4.1.5. Gerenciamento de Teste

O gerenciamento do processo de teste é tão importante quanto à execução dos próprios testes. O sucesso do teste depende de um gerenciamento eficaz. De acordo com Bartram e Stricker (2001), o gerenciamento do teste tem os seguintes objetivos: clareza na estrutura hierárquica das tarefas, planejamento facilitado para o controle de recursos e a realização de tarefas, transparência no uso de métricas, boa compreensão da documentação gerada.

A atividade de teste é considerada caro e demorada. Estes problemas são até mesmo mais aparentes quando o teste é realizado para aplicações *Web*, cujos orçamentos e tempos são muito curtos. Melhoria contínua do processo de teste, segundo Ocampo (1999), é um fator o qual deve ser dada bastante atenção.

4.2. Processo de Teste de Aplicações Web

Para aplicações Web, o processo de teste deve levar em conta à própria estrutura da aplicação e definir em que momento deve ser executado os testes. Outra questão é quanto à documentação, pois aplicações Web possuem a característica de rápida atualização e necessidade para novos testes. Desta forma, quais as particularidades das aplicações Web que devem ser levados em conta no momento da execução do processo de teste.

Stottlemyer (2001) define um processo para aplicações Web contendo as seguintes atividades: definição dos objetivos da aplicação Web, planejamento das estratégias de realização dos testes, planejamentos dos casos de testes, execução dos testes e obtenção dos resultados. Para cada atividade, é definido um tipo de teste que será realizado. Por exemplo, caso o teste seja de compatibilidade, é necessário especificar os objetivos da aplicação ao teste de compatibilidade, definir um planejamento para a realização dos testes, criar os casos de testes, executar os testes e verificar os resultados relativos ao teste de compatibilidade.

Para Perry (2000), o processo de teste de aplicações Web realiza atividades sobre informações da descrição de tecnologias de hardware e software utilizados na aplicação, como sistemas operacionais, navegadores, tecnologias de sistemas distribuídos, segurança no acesso de aplicações, arquitetura de múltiplas camadas entre

outras. E os resultados das atividades são relatórios dos testes. São quatro as atividades definidas:

- Seleção dos riscos de teste em aplicações Web e incluir no plano de teste;
- Seleção dos testes que devem ser realizados;
- Escolha das ferramentas automatizadas;
- Execução os testes.

Após a execução dessas atividades é necessário verificar se os testes foram realizados corretamente. Caso positivo, deve ser emitido um relatório contendo os resultados obtidos. Se por outro lado, acontecer algum problema, deve ser realizado o processo novamente. As três primeiras atividades envolvem o planejamento dos testes. A principal diferença entre os testes de aplicações Web e outros tipos de testes é unicamente concentrada com os riscos associados com as tecnologias associadas. A Figura 4.1 apresenta o processo de teste de aplicações Web proposto por Perry (2000).

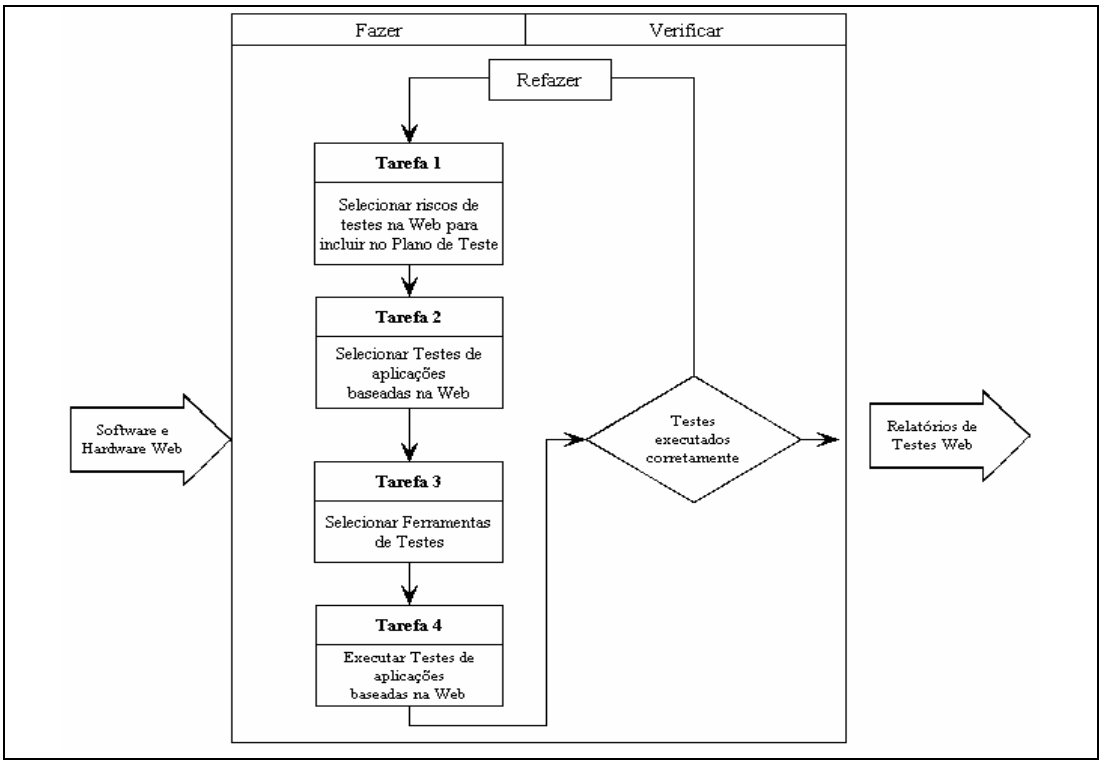


FIGURA 4.1 - Processo de Teste de Aplicações Web

Para Pressman (2000), o teste de aplicações Web deve seguir um processo um pouco diferenciado em relação ao software tradicional. A seguir é apresentada a abordagem recomendada para os testes de aplicações Web quanto ao processo de teste.

1. **O modelo de conteúdo é revisto para descobrir erros.** Esta atividade de teste é similar em muitos aspectos com a revisão de documentos impressos. Um site grande

pode utilizar os serviços de um editor profissional que irá descobrir erros de tipografia e gramática, consistência do conteúdo, representações gráficas, dente outros.

2. **O modelo de projeto é revisto para descobrir erros de navegação.** Cada cenário é exercitado de acordo com o projeto de arquitetura e navegação. Isto serve para encontrar erros de navegação onde o usuário não consegue chegar à página desejada.

3. **Componentes selecionados passam por um processo de teste de unidade.** Nos aplicativos para Web o conceito de unidade muda. Cada página contém conteúdo, links, forms, scripts etc. Nem sempre é possível testar cada uma dessas características individualmente. Em muitos casos, a menor unidade testável é a página. No software tradicional o teste de unidade é focado em detalhes de algoritmo de um módulo e dos dados que fluem pela interface do módulo. Em aplicações Web o teste é focado pelo conteúdo, processamento e links que estão na página.

4. **A arquitetura é construída e testes de integração são conduzidos.** A estratégia para teste de integração depende do projeto de arquitetura que foi escolhida, isto é, de que forma as páginas são interligadas uma as outras.

5. **A aplicação é testada em sua funcionalidade geral e conteúdo.** Assim como no software tradicional, a validação de aplicações Web é focada nas ações do usuário e nas saídas do sistema.

6. **A aplicação é testada quanto a sua compatibilidade com diferentes ambientes.** São definidos todos os prováveis sistemas operacionais, navegadores, plataformas de hardware e protocolos de comunicação. Testes são conduzidos para descobrir erros associados com cada uma das possíveis configurações.

7. **A aplicação é testada por uma população de usuários.** São selecionados grupos de usuários que representem cada tipo de usuário que a aplicação terá. A aplicação é testada para encontrar erros de conteúdo, navegação, compatibilidade e usabilidade.

A abordagem apresentada por Splaine e Jaskiel (2001) para a realização dos testes em aplicação Web é mapeada de acordo com o diagrama do modelo V, como na Figura 4.2.

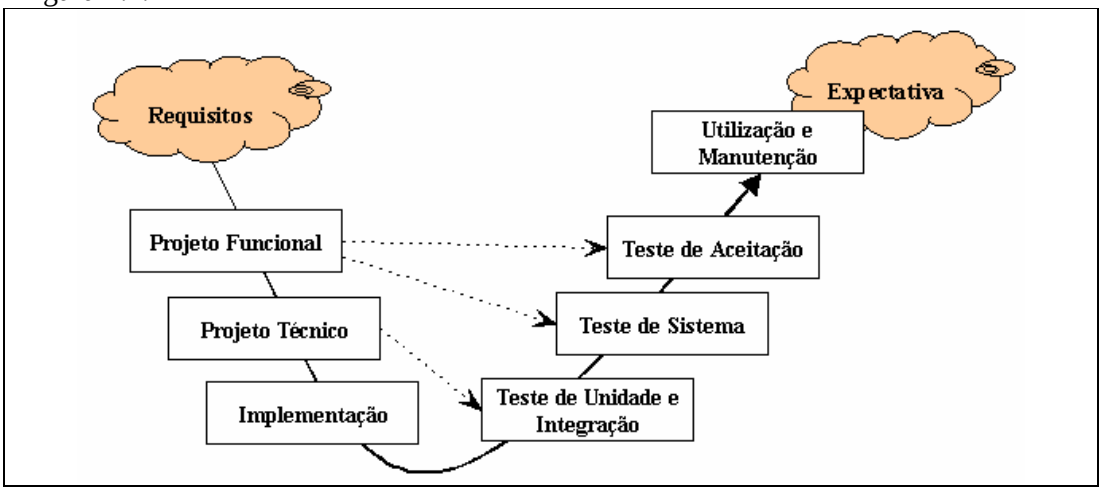


FIGURA 4.2 - Modelo V

Assim, os testes são classificados e realizados de acordo com as fases de teste da Figura 4.2. A ordem de execução do teste é definida da seguinte maneira: garantia de qualidade do código, compatibilidade, navegação, funcionalidade, usabilidade e acessibilidade, desempenho, confiabilidade e segurança.

Assim, para cada fase de realização do teste, são especificados quais os testes que devem ser realizados. Por exemplo, para o teste de unidade e de integração, são executados os testes de validação de código, compatibilidade, navegação e funcionalidade. A tabela 4.1 apresenta a relação dos tipos de testes e as respectivas fases (SPLAINE; JASKIEL, 2001).

TABELA 4.1 - Mapeamento dos tipos e fases de testes

	Teste de Unidade e Integração	Teste de Sistema	Teste de Aceitação
Acessibilidade	✓	✓	
Disponibilidade	-	-	✓
Compatibilidade	✓	✓	✓
Funcionalidade	✓	✓	✓
Navegação	✓	✓	✓
Desempenho	-	✓	✓
Confiabilidade	-	✓	✓
Segurança	-	✓	✓

Como conclusão desta Seção, observou-se que o processo de teste de aplicações *Web* deve ser bem definido e apresentar clareza em todas as suas atividades. Para a implementação da *WebTestManager* foi adotado um processo que abrange as abordagens apresentadas acima. O processo então definido é composto das seguintes atividades:

- ✓ **Definição dos requisitos de teste**: etapa para a especificação do que deve ser testado. Os requisitos são definidos de acordo com os objetivos e os riscos do projeto.

- ✓ **Especificação dos casos de teste**: para qualquer tipo de teste, é necessário especificar quais os casos de teste que devem ser executados.

- ✓ **Seleção das ferramentas automatizadas**: para cada execução dos casos de teste, pode ou não ser realizado com ferramentas automatizadas.

- ✓ **Execução do teste**: nesta etapa os casos de teste são executados e os resultados são armazenados.

- ✓ **Análise dos resultados**: na etapa final, os dados obtidos da execução dos testes são analisados de forma que sejam verificadas as características dos erros.

4.3. Ferramentas Automatizadas

O processo de planejamento de teste pode ser suportado por ferramentas automatizadas também. As ferramentas facilitam o gerenciamento das atividades e um controle das informações do processo de teste. Desta forma, o trabalho da equipe de teste e da equipe de desenvolvimento fica facilitado, pois todo o processo de teste é documentado e as atividades controladas com a finalidade de melhorar o desempenho e organização da atividade de teste.

Para aplicações *Web*, a utilização de tais ferramentas se faz necessária visto que são aplicações que possuem uma grande quantidade de atualizações e como conseqüência exigem novos testes. Outras justificativas são os diversos tipos de testes que devem ser executados, os quais cada um possui uma característica na forma de execução.

Foram estudadas duas ferramentas para planejamento de testes: a TestDirector da Mercury e a e-Manager da Empirix. Essas ferramentas apresentam algumas funcionalidades semelhantes quanto ao gerenciamento do processo de teste. A Tabela 4.2 apresenta um resumo quanto às principais funcionalidades fornecidas, extraídas dos seus manuais.

TABELA 4.2 - Comparação de ferramentas

Funcionalidades	TestDirector	e-Manager
Definição de requisitos de teste	Sim	Sim
Definição de casos de teste	Sim	Sim
Apresentação de resultados	Sim	Sim
Integração com outras ferramentas	Sim	Sim
Suporte ao teste funcional e desempenho	Sim	Sim
Suporte ao teste de compatibilidade e navegação	Não	Não
Suporte ao teste de segurança	Não	Não
Suporte a reparo de erros	Sim	Não
Acompanhamento dos testes	Sim	Não

Conforme listada na Tabela 4.2, as duas ferramentas apresentam um bom suporte para o planejamento dos testes. Podem-se definir os requisitos, casos de teste e gerar relatórios com informações do planejamento. Existe a possibilidade de integração com outras ferramentas de teste funcional e de desempenho. Porém, são restritas a ferramentas do próprio fornecedor. Desta forma, outros tipos de testes não são suportados por elas. A ferramenta TestDirector apresenta funcionalidades a mais do que a e-Manager, como acompanhamento dos testes e suporte de reparo aos erros encontrados.

Assim, verificou-se que as ferramentas de apoio ao suporte ao processo de aplicações Web disponíveis não suportam gerenciar outros tipos de testes como navegação, segurança, compatibilidade, dentre outros. Além do mais, cada fornecedor só apresenta integração com suas ferramentas, não sendo possível integração com ferramentas de outros tipos de teste de outros fornecedores.

Outras ferramentas de teste para aplicações Web foram encontradas em Ricca e Tonella (2001-b), Niese (2002) e Margaria (2002). Essas ferramentas são direcionadas exclusivamente para o teste funcional, com algumas particularidades quanto ao teste estrutural. Outras abordagens são feitas quanto ao fluxo de dados e também quanto ao teste estrutural em Kung (2000-a) e Kung (2000-b).

4.4. Conclusões

O processo de teste de software deve possuir fases onde é descrito o que deve ser testado e como esses testes são realizados (PERRY, 2000). O processo deve suportar a documentação do planejamento e os resultados obtidos com os testes. Aplicações baseadas na Web apresentam características que necessitam de diversos tipos de testes.

Desta forma, a utilização de ferramentas automatizadas para o processo de teste de aplicações para Web apresenta inúmeras vantagens para a equipe de teste, pois elas possibilitam gerenciar muitas atividades de teste e permitem documentar os resultados dos diversos testes realizados. Algumas ferramentas automatizadas suportam o processo de teste e realizam a integração com outras ferramentas de teste. Porém, são limitadas a alguns tipos de teste.

5. *WebTestManager:* Ferramenta de Apoio ao Processo de Teste de Aplicações Web

No presente capítulo é apresentada a proposta desta dissertação, que é a modelagem e implementação do protótipo de uma ferramenta de *software* de apoio ao planejamento e gerenciamento do processo de teste de aplicações *Web*, chamada *WebTestManager*. Nos capítulos anteriores, foi realizado um estudo dos tipos de teste e como devem ser realizados testes voltados para Aplicações *Web* e *Web Sites*. Assim, foram identificadas algumas questões, como:

- Diversos tipos de teste devem ser suportados por ferramentas automatizadas, pois geram uma grande quantidade de informações sobre os resultados obtidos;

- Nem todos os tipos de testes são integrados com ferramentas automatizadas de planejamento de teste;

- Falta de um gerenciamento dos resultados dos testes realizados, mostrando o comportamento da aplicação em relação à qualidade da aplicação desenvolvida.

Desta maneira, a ferramenta *WebTestManager* tem como objetivo integrar os módulos de planejamento, execução e resultados dos testes, através do apoio de ferramentas automatizadas voltadas para o teste de aplicações *Web* disponibilizadas no mercado ou mesmo de forma manual.

Essa integração permite armazenar e gerenciar informações dos resultados obtidos dos testes, de acordo com os requisitos e casos de testes, permitindo assim, obter-se visualização e controle de métricas de teste para aplicações de software baseadas na *Web*. Além de disponibilizar a integração e o gerenciamento de ferramentas de diferentes fabricantes, a *WebTestManager* permite o planejamento dos testes de forma manual, ou seja, disponibiliza *interfaces* de entrada de dados de resultados que não sejam obtidos por meio de ferramentas automatizadas.

As integrações com ferramentas de software específicas de teste voltadas para aplicações Web apresentam a seguintes vantagens:

- Análise dos resultados: avaliação dos resultados dos diversos tipos de testes realizados.

- Redução de tempo: reduz o tempo necessário para o planejamento da validação, verificação e do teste de aplicações *web*.

- Melhor acompanhamento de projetos: permite a toda a equipe de desenvolvimento e de teste acompanhar o andamento e resultados dos testes realizados.

- Reusabilidade: promove a reusabilidade de casos de teste devido à localização centralizada de armazenamento de todas as assertivas de casos de testes, incluindo testes manuais ou automatizados.

- Documentação : melhor geração da documentação dos testes.

A ferramenta provê a utilização de ferramentas de teste existentes no mercado de acordo com a especificação de integração definida através de parâmetros. Para cada tipo de teste são definidas quais informações que serão alimentadas para a *WebTestManager*. Essas informações foram verificadas no decorrer do trabalho como sendo as principais utilizadas para a avaliação dos testes e relevantes para o controle de qualidade da aplicação. A integração com essas informações é uma das contribuições que se procura apresentar na elaboração deste trabalho em relação às ferramentas disponíveis mostradas na seção 4.3.

A ferramenta proposta buscou seguir um processo efetivo de teste de aplicações *Web* que apresente as seguintes características de acordo com Ricca e Tonella (2002):

- **Clareza**: para um processo ser efetivo, passos dentro do processo e os resultados do processo devem ser entendidos facilmente. Além disso, todos na equipe de desenvolvimento e controle de qualidade devem uma ter interpretação clara do processo com o objetivo de validar os resultados.

- **Repetitível e Portável**: um processo de teste deve atender as mudanças das aplicações, de versões e de pessoas. Mas um bom processo pode ser adaptado por outras equipes de teste e para novas situações. O investimento na adoção de um processo de teste tem retorno na sua utilização por muitos projetos.

- **Transparência:** o processo deve ser transparente não somente para a equipe de desenvolvedores e de testadores, mas também para outras pessoas que desejam obter resultados para outros processos dentro da organização.

- **Persistência**: o processo deve criar saídas que não são destruídas no final do processo. Em um ambiente de teste, tais saídas são os documentos, planos de testes, resultados de testes e defeitos encontrados. Essas informações devem ser armazenadas em um repositório de dados disponíveis por toda a equipe de teste e desenvolvimento.

- **Baixo *Overhead* versus Benefícios**: finalmente, um bom processo deve possuir pouca interação com os usuários, em troca dos benefícios que podem ser obtidos dos resultados do processo.

Desta forma, a *WebTestManager* foi projetada com o objetivo de disponibilizar um processo de teste que atenda às necessidades de aplicações *Web* e das características acima através de quatro atividades principais: Levantamento de Requisitos, Definição de Casos de Teste, Execução dos Testes e Análise dos Resultados. Essas atividades procuraram ser adaptadas e desenvolvidas especificamente para as particularidades inerentes a *Web Sites* e Aplicações *Web*.

Este capítulo apresenta a modelagem e as funcionalidades da ferramenta desenvolvida. A seção 5.1 apresenta detalhes da arquitetura proposta para a ferramenta. Na seção 5.2, a modelagem da ferramenta é apresentada. As funcionalidades da ferramenta são mostradas na seção 5.3, e finalmente, na seção 5.4 são abordadas características de integração com ferramentas automatizadas. As conclusões do projeto da ferramenta são mostradas são na seção 5.5.

5.1. Arquitetura da Ferramenta

A ferramenta proposta possui três módulos que permitem uma melhor decomposição das funções a serem suportadas no processo de teste. Cada módulo fica responsável por uma tarefa específica, possibilitando também a adição de novos módulos de acordo com a necessidade de ampliação da ferramenta.

Para o armazenamento de informações, a ferramenta suporta os seguintes bancos de dados: Projetos, Requisitos de Teste, Casos de Testes, Ferramentas de Teste e Resultados. Os três módulos principais são responsáveis pelas funcionalidades principais que devem suportar o processo de teste. A Figura 5.1 apresenta a arquitetura da ferramenta, considerando que as ferramentas específicas são produtos de *software* automatizados do mercado.

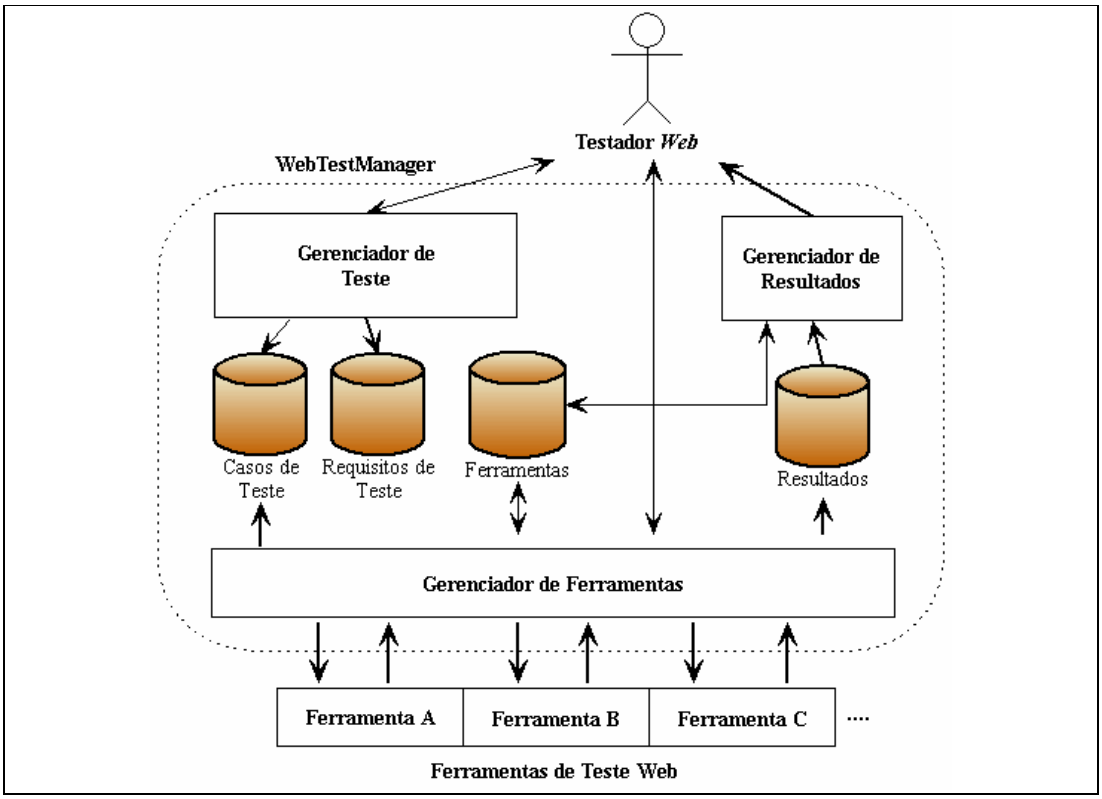


FIGURA 5.1 - Arquitetura da Ferramenta WebTestManager

5.1.1. Gerenciador de Teste

O módulo Gerenciador de Teste é responsável por todo o planejamento e visualização do andamento das atividades no processo de teste. As especificações dos requisitos e casos de testes são realizadas de acordo com o planejamento que foi definido para o respectivo projeto a ser testado. Um Requisito de Teste é uma formalização do que testar e quando testar.

O planejamento do teste é realizado em função dos critérios necessários para a realização dos testes. Os critérios são definidos de acordo com os seguintes testes a serem executados: validação do código, testes de links, testes de segurança, testes de desempenho, testes de compatibilidade, testes funcionais e mais alguns aspectos de

qualidade. O planejamento é essencial para definir um conjunto de testes de acordo com o objetivo e a complexidade da aplicação.

As especificações de requisitos e os casos de teste são armazenadas em um repositório de dados com o objetivo de ser compartilhado por desenvolvedores e testadores. A execução dos casos de teste permite definir o modo que o teste será realizado, podendo ser manual ou automático.

O Gerenciador de Teste controla o andamento do processo de teste disponibilizando um fluxo de atividades de forma planejada e gerenciável, contribuindo assim para um melhor controle da fase de teste. Portanto, o gerenciador de teste, tem como principais funções:

- Realizar a comunicação com usuário para o cadastro de projetos, requisitos de teste, casos de teste, planos de teste, etc;
- Fazer ativação de ferramentas adequadas de acordo com o gerenciador de ferramentas;
- Apresentar o andamento das atividades de um determinado projeto de teste.

5.1.2. Gerenciador de Ferramentas

A integração dos resultados das ferramentas automatizadas é a característica mais importante da *WebTestManager*. O módulo Gerenciador de Ferramentas tem como objetivo permitir a integração da *WebTestManager* com outros produtos de *software* que realizam testes específicos para aplicações *web*. A ferramenta proposta apresenta uma camada de *interface* com algumas ferramentas automatizadas de forma que os resultados sejam integrados com a finalidade de centralizar os dados dos testes realizados. Para cada tipo de teste definido no planejamento e requisitos de teste, o gerenciador ativa uma ferramenta para a realização do teste. Após a execução, o módulo gerenciador obtém os resultados gerados pela respectiva ferramenta que executou o teste.

O gerenciador permite efetuar também o cadastro de ferramentas, a forma de integração e os parâmetros que serão utilizados quando da execução da respectiva ferramenta. A integração pode ser realizada de forma automática, isto é, a *WebTestManager* fará a leitura de um arquivo contendo os resultados dos testes obtidos, ou será disponibilizada uma *interface* de entrada dos dados previamente definidos para o referido teste e informados pela ferramenta.

Para que todas as necessidades de integração com outras ferramentas de teste *web* sejam atendidas, o gerenciador de ferramentas apresenta as seguintes funcionalidades:

- **Registro, Remoção e Alteração das Ferramentas**
  O gerenciador deverá permitir que o testador possa registrar, remover e alterar informações das ferramentas que serão utilizadas para a Execução dos Testes de acordo com os Requisitos de Testes.

- **Registro das Ferramentas às Atividades de Teste**
  Após o registro das informações das ferramentas, o gerenciador deverá permitir ao testador a possibilidade de relacionar as ferramentas com as atividades de testes que se destinam executar a fim de que se tenha um relacionamento lógico entre as

atividades e suas ferramentas. Por exemplo, o teste de *links* deve possuir um relacionamento com as ferramentas registradas.

- **Disponibilizar Visibilidade das Ferramentas Registradas**
  Registradas as ferramentas, o gerenciador deve permitir que o testador possa verificar as ferramentas alocadas para a Execução dos Testes e ter acesso direto a elas.

- **Ativar Ferramentas**
  O gerenciador deve permitir a ativação das Ferramentas segundo condições para Execução dos Testes e o *log* mantido sobre o uso das Ferramentas. Ao final do uso das Ferramentas, o sistema deve certificar que o arquivo resultante foi gerado.

- **Gerar *Log* de Uso das Ferramentas**
  Ao final do uso das Ferramentas, o Gerenciador deve gerar e manter um Histórico contendo as informações resultantes desta execução a fim de que esse possa servir, a *posteriori*, como um arquivo de validação da Execução de Teste.

- **Permitir a Execução do Teste**
  O Gerenciador de Ferramentas deve permitir a Execução dos Testes através do uso das Ferramentas alocadas. Para tal, o gerenciador deve observar todos os fluxos, pré-requisitos e regras inerentes à efetivação do uso de cada ferramenta, bem como a checagem de informações pré-definidas ou geradas ao longo da execução.

5.1.3. Gerenciador de Resultados

O terceiro módulo da *WebTestManager* é o Gerenciador de Resultados que tem como objetivo apresentar os diversos resultados gerados pelos testes, de forma que esses resultados possam ser gerenciados na medida que ocorra a execução do teste ou mesmo para comparação com outras versões do teste. O aspecto importante é o acompanhamento dos resultados, que é a tarefa de visualizar características dos testes em relação às versões de execução, facilitando assim, a visualização temporal dos testes.

Outra função importante do Gerenciador é integrar os resultados obtidos dos diferentes tipos de Testes. Por exemplo, no teste de *Links* (Navegação), pode aparecer um *link* com erro o qual é apresentado também no Teste Funcional. Desta forma, podem-se visualizar informações relacionadas aos dois testes. Outra questão é a análise dos resultados em relação às métricas de Teste. Com a integração de vários tipos de testes, será possível trabalhar com análises das informações obtidas com as ferramentas.

5.2. Modelagem

A modelagem da *WebTestManager* foi especificada de acordo com os requisitos levantados para o processo de teste de aplicações *Web*, abordado neste trabalho no capítulo anterior. Nas seções seguintes são apresentados os Diagramas de Casos de Uso e Diagrama de Classes da ferramenta, em notação UML (JACOBSON et. Al., 2000). O Diagrama de Casos de Uso apresenta as principais funcionalidades que devem ser implementadas pela ferramenta. A visão estática da modelagem é apresentada através

do Diagrama de Classes. Os outros diagramas, os Diagramas de Interação e Diagramas de Atividades são apresentados no decorrer e Anexo 1 do trabalho, respectivamente.

5.2.1. Casos de Uso

Os casos de uso analisados são mostrados na Figura 5.2. Os casos de uso seguem também as principais atividades a serem realizadas no processo teste de aplicações *Web*. Os atores principais que utilizam a ferramenta são definidos como o desenvolvedor da Aplicação e o Testador. Os dois atores participam de dois casos de usos semelhantes. O caso de uso Definir Requisitos de Testes, tem como objetivo o cadastro das definições dos requisitos de testes a serem realizados. Já o caso de uso Visualizar Resultados tem como finalidade permitir que o desenvolvedor e Testador tomem conhecimento dos resultados obtidos com os Testes.

O caso de uso Cadastrar de Projeto tem como objetivo cadastrar informações referentes ao projeto que será testado. Cada Aplicação *Web* ou *Web Site* será considerado como um projeto para fins de teste.

Para o caso de uso Definir Casos de Teste, são especificados os casos de testes de acordo com os requisitos que foram definidos. Estes requisitos são definidos pelo caso de uso Definir Requisitos de Teste, que tem por objetivo que o testador forneça as principais informações referentes aos requisitos de teste do projeto.

O caso de uso Executar Testes tem como objetivo executar os testes que foram definidos nos casos de testes. Neste caso de uso estão incluídos os casos de uso dos vários tipos de testes que podem ser implementados para uma aplicação *Web*. Neste trabalho serão considerados cinco casos de uso para o teste, de acordo com as principais áreas consideradas na Figura 3.4: navegação (*links)*, desempenho, compatibilidade, funcionalidade e segurança.

De acordo com objetivo e requisito de teste pode-se definir, através do caso de uso Planejar de Teste, quais serão os planos de testes que deverão ser executados.

O caso de uso Definir de Parâmetros de Ferramentas permite o testador fixar a forma que será feita a integração de ferramentas com a *WebTestManager*, em relação aos dados. Já o caso de uso Registrar Ferramentas permite adicionar informações referentes a uma ferramenta específica. Finalmente, Coletar Métricas de Teste, é o Caso de Uso na qual será feita uma análise em função dos resultados de testes obtidos.

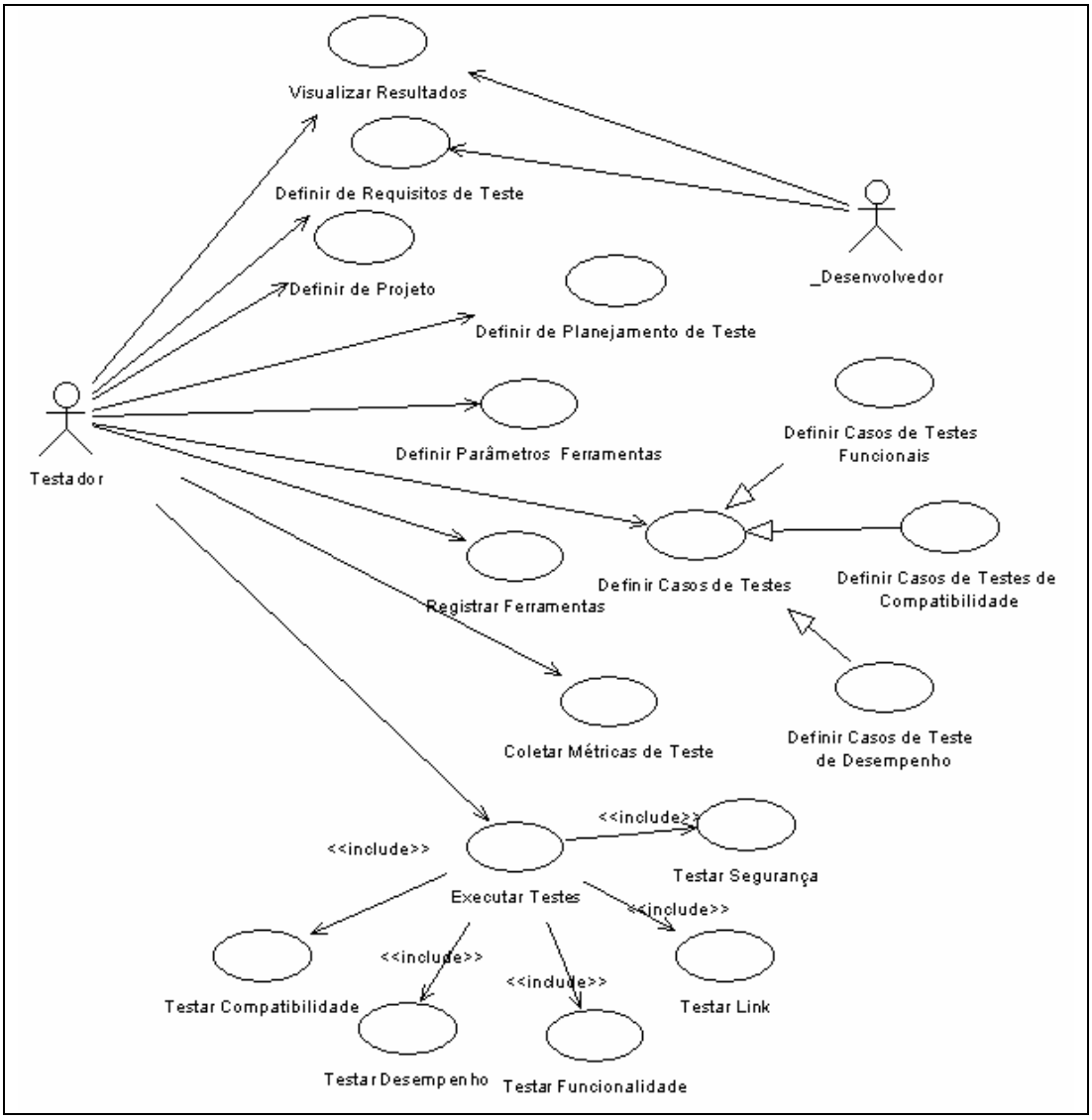


FIGURA 5.2 - Diagrama de Caso de Uso

Os casos de uso descritos para a *WebTestManager* apresentam algumas funcionalidades que não são cobertas por outras ferramentas de planejamento de teste voltadas para Aplicações *Web*. Desta maneira, uma ênfase maior será dada a essas funcionalidades como forma de justificar o desenvolvimento deste trabalho.

5.2.2. Funcionalidades da Ferramenta

De acordo com o diagrama de casos de uso, a ferramenta *WebTestManager* desenvolvida implementa as seguintes funcionalidades:

- Definição de Projetos de Teste

A função do cadastro de projetos permite um melhor controle do andamento do teste para cada projeto. Aplicações *Web* apresentam um grau de atualização e modificação muito grande, assim existe a necessidade de acompanhamento da garantia de qualidade dos projetos. A classe Projeto possui os métodos que implementam esse gerenciamento.

- Definição de Requisitos de Teste

A especificação dos requisitos de teste permite definir o que deve ser testado na aplicação. Essa definição é importante visto que aplicações *Web* implementam uma grande variedade de funcionalidades (GRAHAM, 2002).

- Definição de Casos de Teste

Uma importante funcionalidade é quanto à definição dos casos de teste. Como visto nos capítulos anteriores, aplicações *Web* necessitam de vários tipos de teste, cada um podendo conter diversos casos de testes.

- Execução de Teste

A execução dos testes tem como função executar os casos de teste para os tipos de testes de aplicações *Web*. A execução é concretizada com a ativação de uma ferramenta ou com a entrada dos resultados do teste. Os principais testes são:

- ✓ Teste de Compatibilidade;
- ✓ Teste Funcional;
- ✓ Teste de Desempenho;
- ✓ Teste de Navegação.

- Registrar Ferramentas

Uma função importante dentro da *WebTestManager* é o registro de ferramentas que são utilizadas para a execução do teste. O armazenamento das características das ferramentas permite um melhor acompanhamento no uso das ferramentas e também na questão da forma de integração com os resultados gerados.

- Analisar Resultados

A visualização dos resultados de todos os testes de forma integrada tem como finalidade um melhor gerenciamento das informações geradas pelas ferramentas de teste. Outra função é a análise dos resultados em relação às várias versões de teste executadas.

Com o objetivo de apresentar uma descrição do cenário de execução das principais funcionalidades, a Figura 5.3 apresenta o diagrama de seqüência, no qual são descritas as principais classes e suas interações com o objetivo de executar o teste de aplicações *Web*.

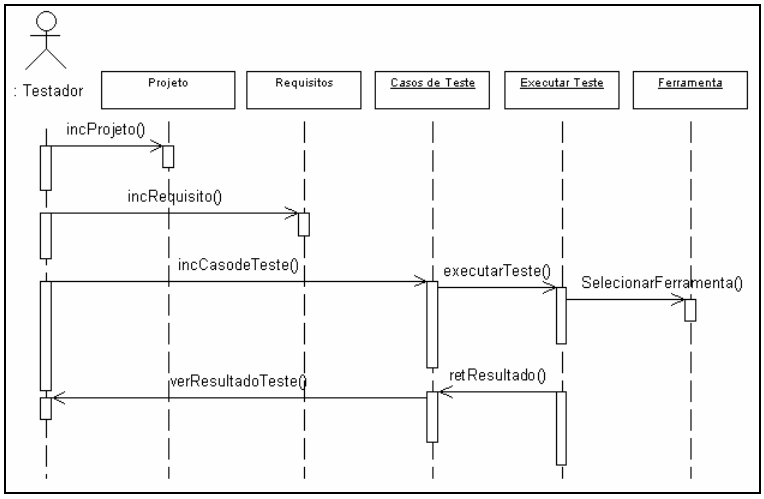


FIGURA 5.3 - Diagrama de Sequência

5.2.3. Diagrama de Classes

Ao analisar quais os principais testes a serem desenvolvidos, os passos do processo de teste e informações referentes das ferramentas, definiu-se o seguinte diagrama de classes da *WebTestManger*, apresentado na Figura 5.4.

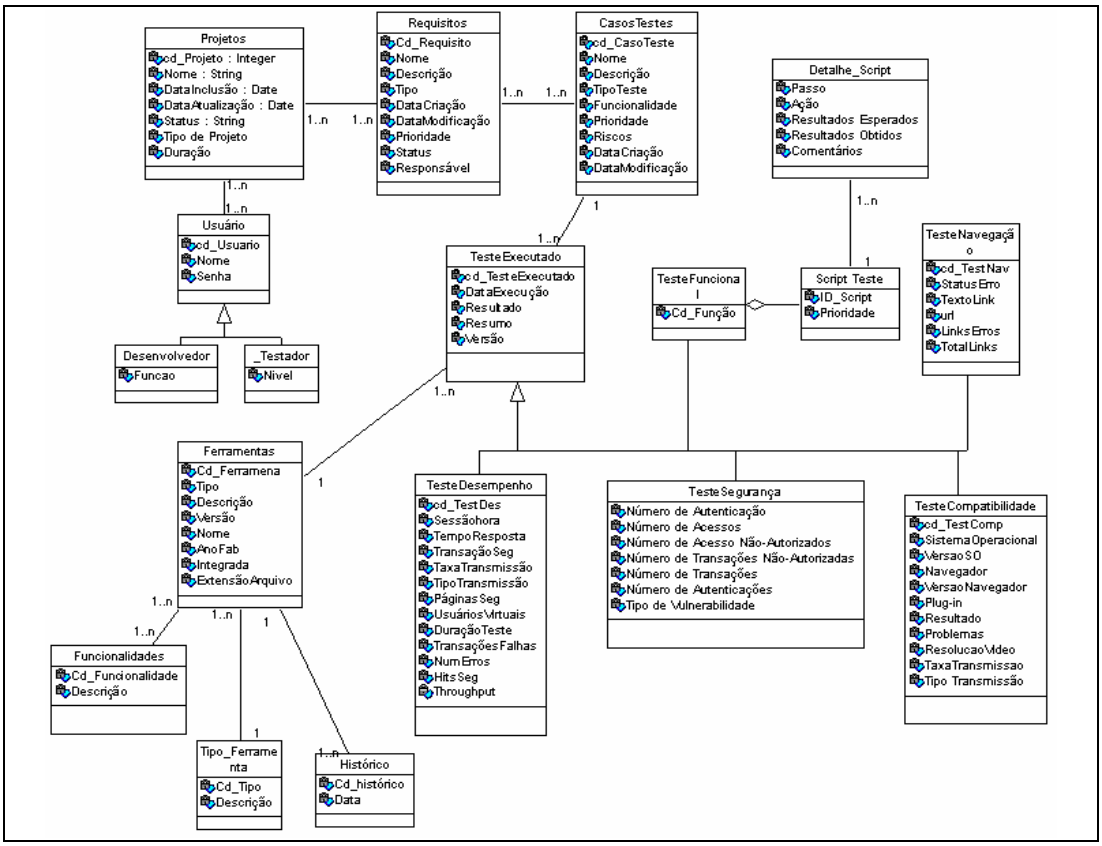


FIGURA 5.4 - Diagrama de Classes

5.2.3.1 Classe Projeto

Na classe Projeto, são definidos os parâmetros de informações para uma determinada aplicação *Web*. A classe Projeto pode conter vários usuários, desenvolvedores ou testadores. Um atributo importante da classe é o tipo de projeto, pois indica qual categoria de aplicações *Web* o projeto está inserido. Isso é importante para um levantamento dos resultados de teste e da relação com o tipo de projeto.

5.2.3.2 Classe Requisitos de Teste

Essa classe é responsável por prover a definição e gerência dos requisitos de teste para um projeto específico. Usuários podem especificar detalhes de requisitos como: *status* de cada requisito, tipo, prioridades e uma breve descrição.

5.2.3.3 Classe Casos de Teste

A classe Casos de Teste serve para armazenar casos de teste contendo informações fornecidas pelo testador ou de forma automática por uma ferramenta do tipo C/R, que serve para o teste funcional e de desempenho. Essas ferramentas fornecem um *script* de teste de acordo com a ação do usuário com a aplicação. Porém, existe a possibilidade de entrar com os passos de teste para montar o *script* de teste de forma manual.

A classe Casos de Teste possui as classes especializadas para a definição de cada tipo de teste, por exemplo, casos de teste de compatibilidade ou casos de teste de navegação. Assim, podem ser definidos casos de teste que não são suportados por outras ferramentas de planejamento de teste existentes como mostrada na Seção 4.3.

5.2.3.4 Classe Teste Executado

A classe Teste Executado serve para que o teste definido através dos casos de teste seja executado de acordo o seu tipo e a ferramenta que venha ser utilizada. A classe Ferramenta é relacionada com esta classe, pois na execução do teste, a *WebTestManager* utiliza uma ferramenta automática de teste. A classe Teste Executado possui cinco classes especializadas, que são responsáveis por cada tipo de teste considerado neste trabalho.

5.2.3.5 Classe *TesteCompatibilidade*

A Classe *TesteCompatibilidade* é responsável por acompanhar e armazenar os resultados dos testes de compatibilidade realizados sobre a aplicação. A classe possui os seguintes atributos: tipo de processador, taxa de transmissão e informação da resolução do monitor de vídeo como informações sobre características de *hardware*. Já para informações inerentes a *software*, a classe contém os seguintes atributos: sistema operacional, versão do sistema operacional, navegador (*browser*), versão do navegador e *plug-in* suportado. O atributo Resultado armazena a informação se o teste passou ou se ocorreu algum tipo de problema. Caso tenha ocorrido algum tipo de problema, a descrição do mesmo é armazenada no atributo Problema.

5.2.3.6 Classe *TesteDesempenho*

Esta classe armazena e trabalha com informações dos resultados do teste de desempenho com o objetivo de obter relações entre os diversos parâmetros dos resultados do testes. A partir das especificações dos casos de teste de desempenho, a classe permite uma análise em cima dos resultados obtidos para um determinado processo de negócio da aplicação a ser testada o desempenho.

5.2.3.7 Classe *TesteFuncional*

A classe *TesteFuncional* tem como objetivo definir as informações quanto à realização dos testes funcionais da aplicação. A classe trabalha com classes de *scripts* de teste que indicam os passos necessários para a realização de uma determinada atividade sobre a aplicação. As informações dos *scripts* são repassadas para as ferramentas automatizadas que têm a responsabilidade de executá-los.

5.2.3.8 Classe *TesteNavegação*

As informações em relação ao teste de navegação (*links*) estão contidas nesta classe. Porém são informações relacionadas somente com os resultados obtidos. Assim, a classe armazenará informações das várias versões dos testes de *links* realizados, permitindo também descobrir informações ligadas com os outros tipos de teste.

5.2.3.9 Classe Ferramenta

Classe ferramenta define as especificações das ferramentas automatizadas que são integradas a *WebTestManager*. A classe permite o gerenciamento das ferramentas através da manutenção de *logs* de utilização. Outra importante função é a definição dos parâmetros de integração com as ferramentas.

5.3. Implementação das Funcionalidades

Nesta Seção é apresentada a implementação do protótipo com as principais funcionalidades. A *WebTestManager* foi desenvolvida no ambiente de desenvolvimento Microsoft Visual Basic 6.0 utilizando o gerenciador de banco de dados Microsoft Access 2000.

Para qualquer teste realizado de acordo com o planejamento, são utilizadas ferramentas automatizadas de teste. Para a utilização da *WebTestManager* é necessário que a empresa possua pelo menos uma ferramenta automatizada para cada teste e que seja integrada à ferramenta proposta. Para o desenvolvimento deste protótipo, foram consideradas no máximo três ferramentas de cada tipo de teste, em alguns tipos foi considerada somente uma ferramenta.

Outro aspecto implementado pela *WebTestManager* é o gerenciamento de utilização das ferramentas. Isso permite ter um controle por parte da equipe de controle de qualidade, do uso das ferramentas nos respectivos projetos. Essa foi uma das funcionalidades não encontradas em outras ferramentas de gerenciamento de teste. Além do mais, essas ferramentas utilizam somente ferramentas de software de sua autoria, não permitindo integrar ferramentas de uso livre ou de outros fabricantes.

De acordo com a arquitetura da ferramenta, mostrada na seção 5.1, *WebTestManager* possui três módulos gerenciadores: o de **Teste**, responsável pelo

planejamento do teste como de definição de casos de teste, o gerenciador de **Ferramentas**, que focaliza a integração e gerenciamento de ferramentas utilizadas e o módulo gerenciador de **Resultados** que trabalha com os dados dos resultados gerados pelas ferramentas de teste ou os testes realizados de forma manual. As métricas de teste para aplicações *Web* podem ser obtidas através das informações do gerenciador de resultados

A seguir são apresentadas as principais funcionalidades da ferramenta, de acordo com a especificação dos casos de uso e o detalhamento no inicio desta Seção, através da forma de implementação através das *interfaces* com o usuário. A próxima seção mostra a estratégia de integração das ferramentas de testes específicos e a *WebTestManager*, como por exemplo, de que forma as ferramentas de teste de desempenho são integradas e quais as estratégias para adicionar novas ferramentas.

5.3.1. Cadastro de Projetos

Para a utilização da ferramenta, será necessário primeiramente especificar um Projeto de Teste *Web*. Este cadastro de projeto permite levantar informações de métricas de teste de acordo com o tipo declarado do projeto. De acordo com Powell (1998), existe uma classificação de software baseado na *Web* como apresentado da Figura 2.4. Assim, em um Projeto *Web* que seja apenas de caráter Institucional, com páginas estáticas, será necessário apenas o teste de navegação, ou seja, a verificação de *links* entre as páginas. A figura 5.5 apresenta a tela para inserir um Projeto na *WebTestManager*.

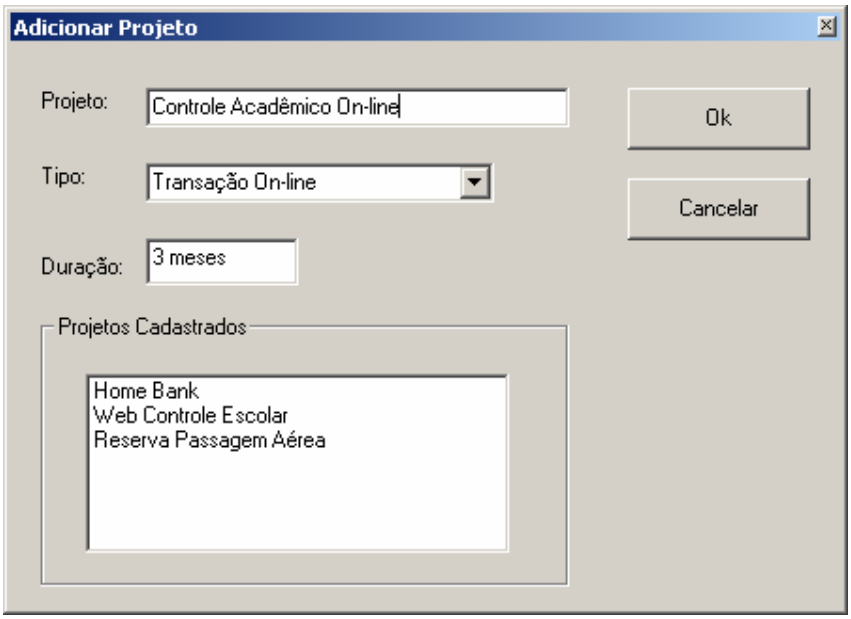


FIGURA 5.5 - Cadastro de Projetos

5.3.2. Adicionar Requisitos de Teste

O processo de teste não é linear e naturalmente difere dependendo das práticas de cada organização e metodologia empregada. Porém, os princípios básicos de todo processo são os mesmos. Assim uma das primeiras atividades no processo de teste é a definição dos Requisitos, ou seja, o que testar e o que a aplicação deve fazer. Desta

forma, o desenvolvimento e gerenciamento de Requisitos de teste são atividades importantes no desenvolvimento e teste de software. Sendo uma Aplicação Web, a preocupação é maior devido à implementação de uma variedade de funcionalidades implementadas e os diversos tipos de teste a serem realizados.

A WebTestManager permite que um Requisito de Teste possa ser definido tanto pelo testador como pelo desenvolvedor. Um requisito de teste deve ser especificado também pela equipe de desenvolvimento, pois envolve a especificação das funcionalidades que estão sendo implementadas. Por exemplo, uma aplicação deverá ter uma variedade de transações importantes associadas com os requisitos teste para assegurar que a aplicação esteja funcionando na realidade como planejado. Estes requisitos de testes deverão ser formulados claramente e é extremamente importante que sejam verificáveis ao longo do processo. A Figura 5.6 apresenta a interface que captura Requisitos de teste.

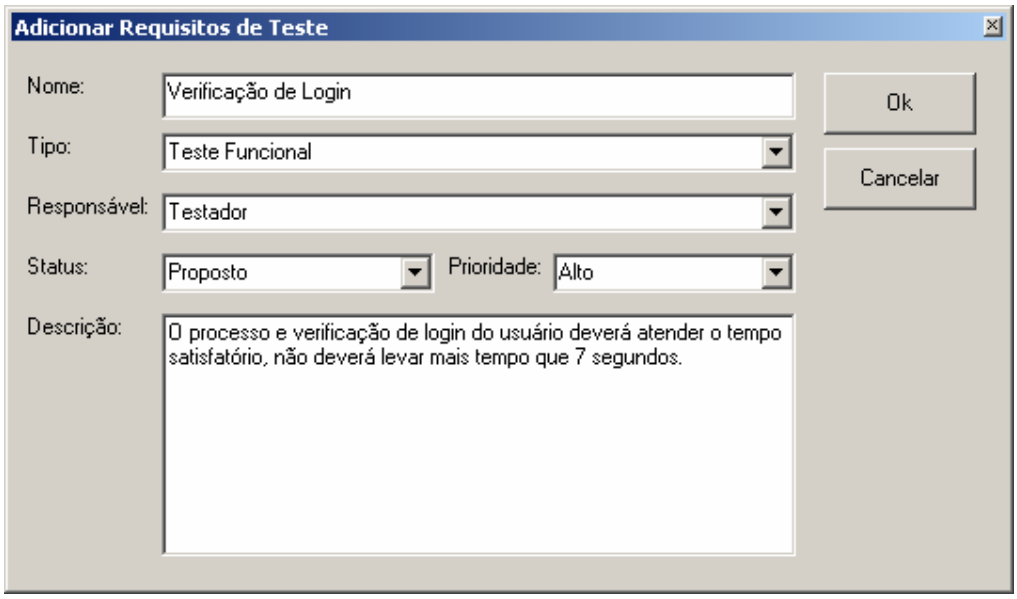


FIGURA 5.6 - Requisitos de Teste

O usuário que estiver especificando um Requisito de teste deverá informar o nome do Requisito, deixando claro seu objetivo. O tipo de teste deverá ser de acordo com os tipos implementados para aplicações *Web*. São disponibilizados testes funcionais, compatibilidade, desempenho, navegação e segurança. De acordo com a escolha do tipo de teste, a ferramenta direcionará para a respectiva configuração dos casos de teste. Deve ser informado também, quem é o responsável pela a entrada do Requisito de Teste, o *status* do andamento do teste: Proposto, Aprovado, Rejeitado, Implementado e Verificado, qual a prioridade: baixa, média ou alta e finalmente uma descrição textual mais detalhada do requisito. A ferramenta armazena automaticamente a data de criação e modificação do requisito.

5.3.3. Cadastro de Casos de Teste

Uma vez criados os requisitos de teste, testadores deverão criar e especificar uma séria de casos de testes que deverão ser executados de acordo com os tipos de teste. Cada Requisito de Teste deverá ter pelo menos um Caso de Teste associado para suportar e em muitos casos um Requisito poderá estar associado a vários Casos de Uso.

Estes Casos de Teste podem ser criados manualmente ou de forma automatizada com auxílio de ferramentas de teste que geram scripts de teste. No caso de testes manuais, os passos de teste devem ser documentados claramente através da especificação de cada passo e o resultado esperado, assim como o resultado obtido no teste.

Casos de Teste deverão ser explicitamente associados a Requisitos de Teste e um Requisito deverá ter pelo menos um caso de teste associado. Para evitar trabalho duplicado, estes Casos de Teste são armazenados em um repositório de dados central onde toda a equipe de teste tem acesso.

Como citado na seção anterior, a ferramenta proposta trabalha com vários tipos de teste voltados para aplicação Web (funcional, compatibilidade, desempenho, etc.), assim para cada tipo serão configurados Casos de Testes que são relevantes para o determinado tipo de Teste. Essa característica da ferramenta permite que se trabalhe não somente com um teste específico, mas com uma variedade de testes de uma forma integrada. Assim, a partir do tipo de teste escolhido no Requisito de Teste, deverão ser especificados, na ferramenta, Casos de Teste que são inerentes ao tipo escolhido. A Figura 5.7 apresenta a tela de entrada de especificação de Casos de Teste funcional da *WebTestManager*.

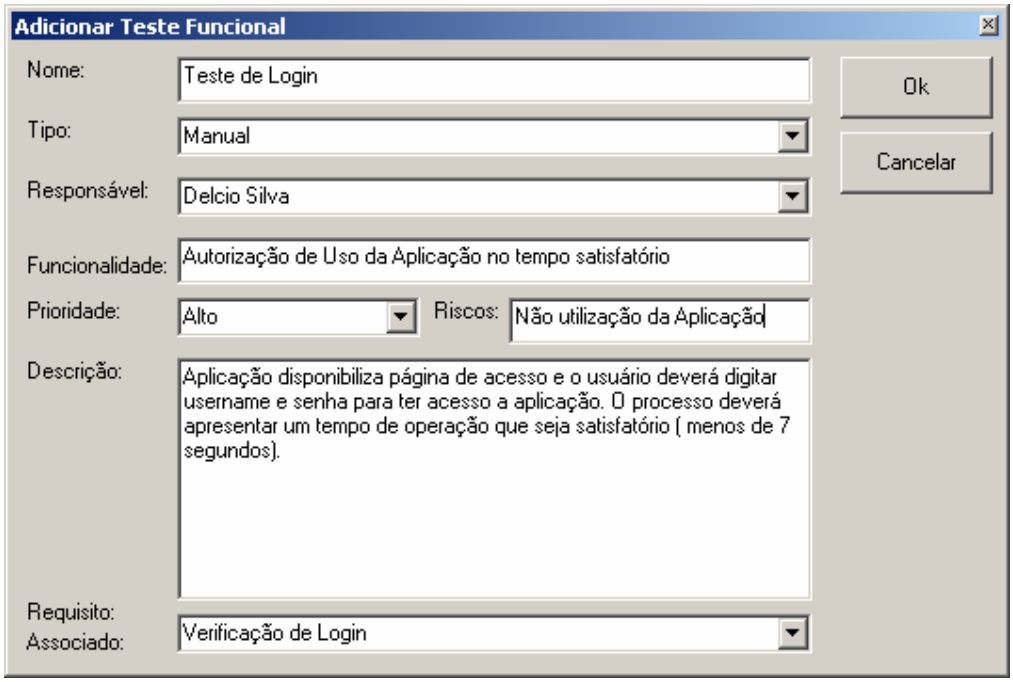


FIGURA 5.7 - Caso de Teste Funcional

O Caso de Teste deve ser especificado no campo Tipo como Manual ou Automatizado. Caso seja feita a opção Manual, os passos de teste deverão ser informados individualmente, como apresentado na Figura 5.7. Caso seja selecionado Automatizado, deverá ser informada qual ferramenta de teste funcional será utilizado para capturar os scripts de teste. Assim, para cada Caso de Teste incluído, é associado um script de teste.

Esses scripts são capturados através de ferramentas do tipo Capture/Replay (C/R), que capturam ações do usuário. No caso de uso dessas ferramentas, os passos de execução do script são importados e são documentados com a ação realizada e os resultados esperados. Para a especificação manual, é necessário que sejam declarados os

passos do script de teste e a execução deve ser feita executando a aplicação manualmente em um navegador (browser). A Figura 5.8 apresenta um exemplo de script de teste capturando o comportamento do usuário.

```
Browser("Mercury Tours").Page("Mercury Tours").Image("Login").Click 66,16
Browser("Mercury Tours").Page("Welcome to Mercury").Image("Search Flights").Click
Browser("Mercury Tours").Page("Find Flights").WebList("depart").Select "Frankfurt"
Browser("Mercury Tours").Page("Find Flights").Image("findFlights").Click 74,11
Browser("Mercury Tours").Page("Search Results").WebRadioGroup("outboundFlight").Select
Browser("Mercury Tours").Page("Search Results").Image("reserveFlights").Click 85,19
Browser("Mercury Tours").Page("Method of Payment").Image("buyFlights").Click 94,19
Browser("Mercury Tours").Page("Flight Confirmation").Image("Book Another").Click 130,8
Browser("Mercury Tours").Page("Find Flights_2").Image("SignOff Button").Click
Browser("Mercury Tours").Page("Mercury Tours_2").Sync
```

FIGURA 5.8 - Script de Teste

A utilização de ferramentas apresenta inúmeras vantagens, pois as ações são gravadas logo após a execução. Porém, caso não seja possível dispor dessas ferramentas automatizadas, fica então a opção da realização manual. A Figura 5.9 apresenta a interface de captura de script de teste de forma manual.

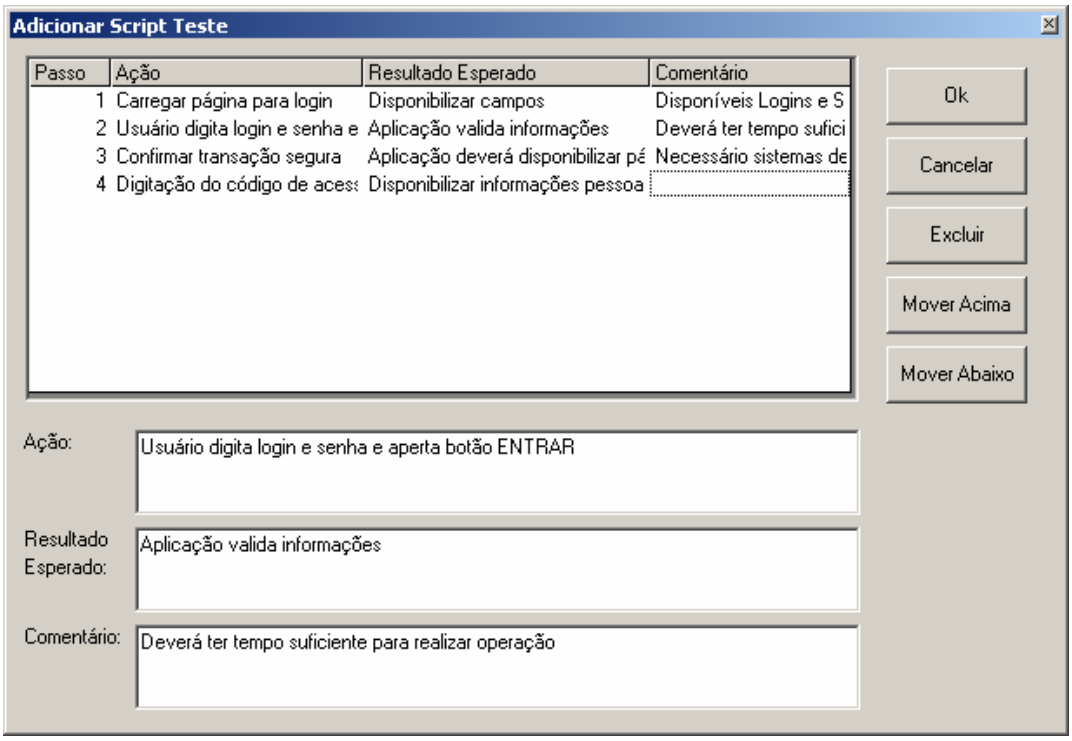


FIGURA 5.9 - Script de Teste Manual

O usuário deverá entrar com os passos que contêm as ações que devem ser realizados na aplicação Web. Essas ações são funções que a aplicação deve executar corretamente de modo que se concretize uma tarefa com sucesso. Para cada passo, deverá ser especificado qual o resultado esperado para a determinada ação. A descrição dos resultados esperados tem que ser declarada de forma objetiva e clara para a ação especificada. Um campo de comentário é disponibilizado com o objetivo de se fazer alguma observação na realização do passo.

A ferramenta proposta neste trabalho permite trabalhar com outros tipos de testes para aplicações Web. Assim, são disponibilizadas interfaces de entrada de dados específicas para a inclusão de casos de teste para os tipos de teste abordados neste trabalho. Desta forma, é possível a especificação de casos de teste classificados como não funcionais. A seguir é apresentado o funcionamento para a especificação dos casos de teste para os testes não funcionais de aplicações Web.

5.3.3.1 Cadastro de Teste de Compatibilidade

Como descrito na Seção 3.2.7, o teste de compatibilidade tem como objetivo encontrar erros ocorridos na utilização da aplicação do lado Cliente. Isso ocorre devido a uma variedade de tecnologias como sistemas operacionais, navegadores (browser), versões de linguagens e suas versões que não permitem que páginas sejam visualizadas corretamente, ou às vezes deixando aplicações sem funcionamento.

A WebTestManager permite que seja especificada, através de casos de teste de compatibilidade, informação das características do ambiente do cliente e das tecnologias empregadas na aplicação. É importante frisar que existem meios de descobrir em tempo de execução, qual o navegador que o cliente está utilizando e outras informações que podem ajudar para que a aplicação seja visualizada sem algum tipo de erro. Porém, para uma aplicação complexa e que tenha como objetivo atingir o maior número de usuários possível, é necessário à realização de um teste de compatibilidade mais elaborado.

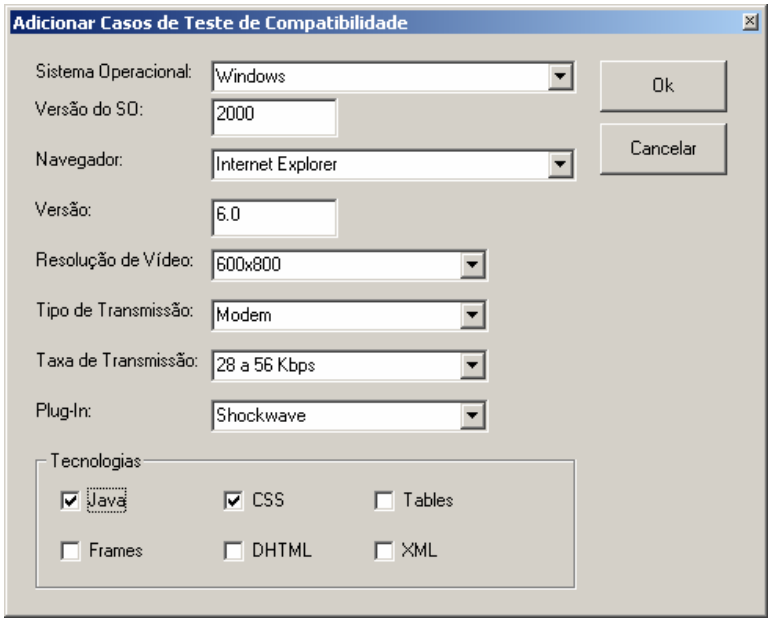


FIGURA 5.10 - Casos de Testes de Compatibilidade

A Figura 5.10 apresenta a tela de entrada para casos de teste de compatibilidade. As informações descritas e que deverão ser informadas foram levantadas junto a Splaine (2001) e Dustin (2001). São informações de casos de teste que devem ser armazenadas com o objetivo de serem utilizadas para futuras versões da aplicação. Essas informações podem ser utilizadas como entradas para ferramentas automatizadas ou usadas no teste interno para a verificação de compatibilidade de acordo com a tabela no Anexo dois. Ou simplesmente armazenadas com objetivo de manter documentado as várias combinações de tecnologias e os problemas apresentados.

5.3.3.2 Cadastro de Teste de Desempenho

Teste de desempenho compreende a verificação se a aplicação Web atende os requisitos de desempenho dependendo do objetivo da aplicação. Os requisitos podem ter várias finalidades como, por exemplo, tempo de resposta, números de usuários simultaneamente ou como uma complexa solicitação do sistema pode suportar dez mil transações por minuto, enquanto continua sendo capaz de carregar uma página no tempo satisfatório (STOUT, 2001). Para realizar este tipo de teste, ferramentas automatizadas são necessárias para gerar carga de trabalho ("workload") e coletar métricas de desempenho da aplicação corretamente.

Existem algumas particularidades diferenciadas dos testes de desempenho de aplicações Web em relação ao software tradicional. A Tabela 5.1 apresenta algumas diferenças em relações aos fatores chave para o teste de desempenho (SAVOIA, 2001).

TABELA 5.1 - Comparação de Teste de Desempenho

Fatores Chave	Teste de Desempenho Tradicional	Teste de Aplicações *Web*
Volume de Transação	Previsível e Limitado	Não previsível e potencialmente ilimitado
Comportamento dos Usuários	Previsível e Controlável	Não previsível e sem controle
Componentes de Sistema	Software e Hardware centralizados, Redes Locais	Roteadores, Firewalls, Hubs, etc.
Riscos	Falhas notadas somente internamente	Falhas altamente visíveis.

Para a realização dos testes de desempenho, é necessário definir algumas especificações dentro de um processo de teste de desempenho, através dos seguintes passos: análise do sistema, criação de Scripts, definição do comportamento do usuário, execução e análise dos resultados (MILLER, 2000). O primeiro passo é uma análise do sistema, que tem como objetivo identificar qual a finalidade e a estratégia de teste, como por exemplo: quais processos/transações a serem testados, números de usuários concorrentes, o total de hits por segundo, etc. O segundo passo é criar Scripts de usuários virtuais, que são usados para capturar todos os processos de negócio na utilização da aplicação. Um usuário virtual emula o usuário real dirigido para uma real aplicação como cliente. Esses Scripts podem ser obtidos através de ferramentas do tipo Capture/Replay.

O terceiro passo é a definição dos comportamentos dos usuários em relação às configurações de tempo de uso de cada usuário virtual. Essas configurações têm por objetivo definir o tempo que cada usuário gastará para realizar uma transação, a velocidade de conexão do usuário e suporte a erros que ocorrerem na aplicação. Assim, é possível criar um escalonamento de uso dos usuários virtuais na aplicação de acordo com a necessidade do teste de desempenho. A criação dos cenários de teste é o quarto passo no processo de teste de desempenho. Os cenários de teste de carga contêm informações sobre o grupo de usuários virtuais que deverá executar os scripts e a definição do número de máquinas que os grupos estão executando.

Para a execução de um cenário de teste com sucesso, o testador deverá primeiramente definir um grupo de usuários virtuais baseado em transações comuns entre os usuários. Num segundo momento, existe a necessidade de definir e distribuir o

total de usuários virtuais, entre os processos de negócios dos usuários para simular múltiplas transações.

A execução dos cenários de teste e o monitoramento do desempenho constituem o quarto passo. O monitoramento em tempo real permite ao testador visualizar o desempenho da aplicação em qualquer tempo durante os testes. Cada componente do sistema requer o monitoramento: clientes, rede, servidor Web, servidor de aplicação, banco de dados e todos os servidores de hardware. E como quinto passo, a análise de resultados tem como objetivo identificar os problemas de desempenho a partir dos dados obtidos nos testes. A análise dos parâmetros é realizada através de gráficos e de relatórios que ajudam a resumir e apresentar os resultados dos testes.

Os casos de teste de desempenho implementados na WebTestManager seguem o processo descrito acima. O objetivo é que as informações de definição dos casos de teste e dos resultados dos testes sejam armazenadas com a finalidade de gerenciamento dos dados obtidos. A parte da execução é de responsabilidade da ferramenta de teste de desempenho selecionada.

Resultante da análise dos principais dados requeridos para os casos de teste de desempenho nas ferramentas investigadas neste trabalho, a interface para entrada dos casos de teste é apresentada na Figura 5.11.

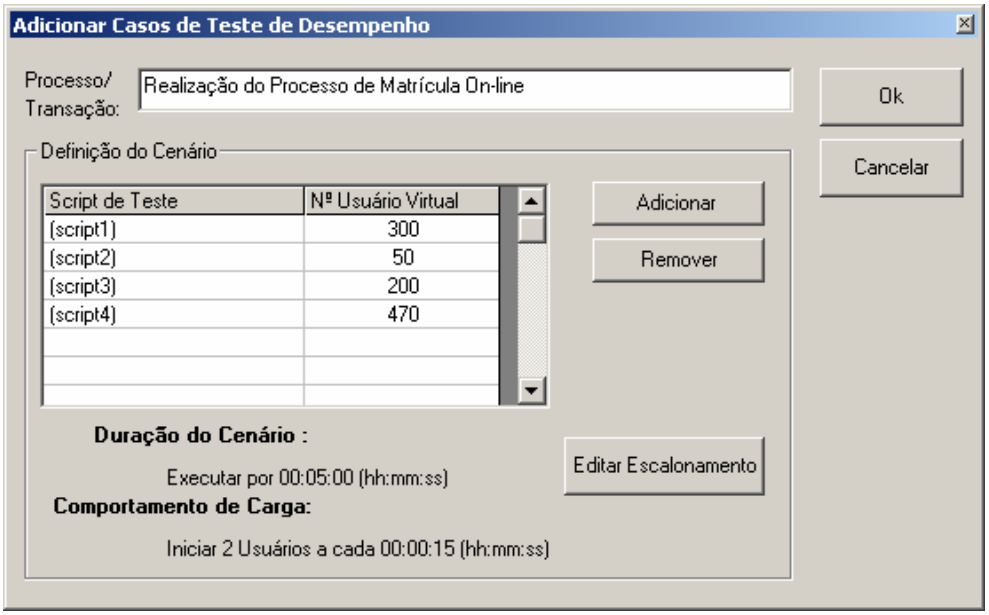


FIGURA 5.11 - Casos de Teste de Desempenho

O testador deverá especificar primeiramente qual o processo/transação do respectivo caso de teste. O objetivo é definir qual o processo de negócio da aplicação do qual será testado o desempenho. Assim, pode-se ter uma visão de como é o comportamento da aplicação perante uma determinada carga. Na definição do cenário de teste, deverá ser informada a quantidade de usuários virtuais para cada script de teste.

Esses scripts podem ser os mesmos criados em decorrência do teste funcional. O testador deverá também configurar o comportamento dos usuários em relação ao escalonamento de uso no tempo. Essa configuração é baseada em termos do inicio do teste, duração e parada. Por exemplo, pode-se definir que todos os usuários iniciarão o teste simultaneamente ou um determinado número a cada intervalo de tempo.

Essas informações de casos de testes serão repassadas para a ferramenta selecionada que irá executar os testes de desempenho. Para este protótipo da

WebTestManager foram utilizadas duas ferramentas para os quais foram estudadas as formas de integração: AstraLoadTest (Mercury Interactive) e e-Load (Empirix). Para outras ferramentas são necessárias especificações para a troca de informações. Em geral, todas que realizam esse tipo de teste trabalham com as mesmas informações e de forma semelhante.

5.3.3.3 Cadastro de Teste de Navegação

Para os testes de navegação, as definições de casos de teste podem ser criadas a partir da documentação de navegação desenvolvida na fase de análise e projeto como: diagramas, matriz e mapas funcionais de navegação (SPLAINE, 2001). Porém, essa documentação indica as ligações entre as páginas de forma conceitual e não implementadas na aplicação real. Assim, para Web Sites e aplicações Web que contém mais do que 100 páginas fica difícil a manutenção da navegação e o tempo consumido é muito grande.

Consequentemente é necessário o uso de ferramentas automatizadas para verificar a ligação entre as páginas, isto é, se a partir de uma página Web A é possível chegar a uma outra página Web B e vice-versa. Porém, essas ferramentas não são capazes de verificar se uma ligação (link) direciona para o conteúdo corretamente. Portanto, este tipo de verificação continuará a ser realizado manualmente.

Os casos de teste de navegação não são realizados como forma de definir o caminho de navegação para cada página, pois o tempo necessário para sua especificação e da periodicidade com que o teste deve ser realizado é muito grande. Na prática, o que ocorre é somente a verificação de forma automática dos links, ou seja, se as ligações funcionam, e informando, como resultados, as páginas Web que contêm links quebrados.

A ferramenta proposta tem como objetivo utilizar os resultados dos testes das ferramentas de forma automática ou inserção de valores dos resultados de maneira manual. A finalidade é manter um histórico dos testes realizados como parâmetro de controle de qualidade e também para a integração com outras métricas dos resultados dos testes obtidos pela ferramenta.

5.3.4. Execução de Teste

O terceiro passo no processo de teste de aplicações Web abordado neste trabalho, é a execução dos testes planejados através dos casos de teste. A execução dos testes deve repassar informações dos casos de teste para as ferramentas registradas. De posse dessas informações, cada ferramenta executa os casos de teste e devolve os resultados obtidos no final da execução. É importante que na etapa de execução, todas as ferramentas específicas de teste estejam registradas e instaladas.

Para cada tipo de teste a ser executado, serão necessárias a seleção dos casos de testes e a ferramenta automatizada quando for a situação. Se por algum motivo, o teste for realizado manualmente, então o testador deverá informar os resultados obtidos para o determinado caso de teste. O testador selecionará qual o tipo de teste que será executado a partir e uma lista dos testes abordados pela ferramenta. A Figura 5.12 apresenta a tela de escolha do tipo de teste.

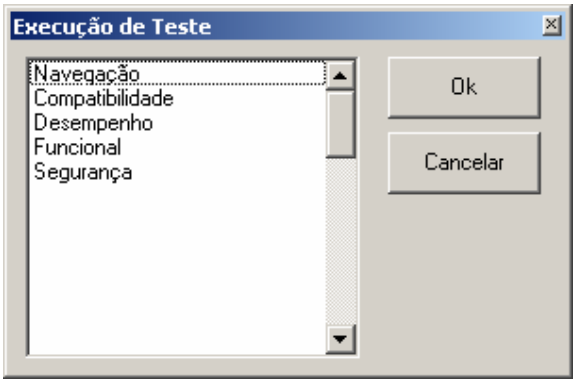


FIGURA 5.12 - Escolha do Tipo de Execução de Teste

A partir da escolha do tipo de teste, será permitido configurar a execução para o determinado tipo de teste. Nessa configuração serão definidos os parâmetros de execução e a visualização dos resultados encontrados. A seguir são apresentadas estas configurações para os testes abordados.

Por ser um teste um pouco complexo, o teste de compatibilidade poderá ser realizado por um dos três processos: manual, automático ou interno. No teste manual, o testador deverá executar a aplicação em diversas plataformas e versões de navegadores, porém fica impossível a realização de todas as combinações possíveis neste caso. Assim, o teste deve ser realizado com as principais tecnologias que são utilizadas no mercado levantadas pelos órgãos de pesquisa.

No processo automático, as informações dos casos de teste serão repassadas para a ferramenta selecionada, a qual ficará responsável de verificar os possíveis problemas de compatibilidade. O terceiro processo é o chamado interno, consiste fazer a verificação a partir de uma tabela de combinações das plataformas e tecnologias utilizadas pela aplicação, ver tabela em Anexo dois. Assim a WebTestManager armazenará todos os casos de testes e seus respectivos resultados. A Figura 5.13 mostra a interface de entrada para a execução do teste de compatibilidade.

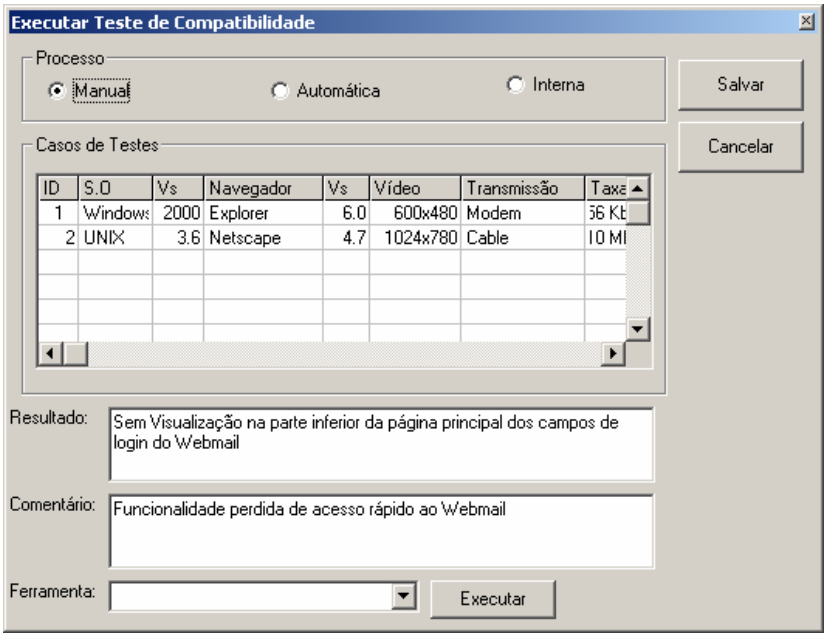


FIGURA 5.13 - Execução do Teste de Compatibilidade

A execução dos testes funcionais se dará pela execução dos *scripts* de testes gerados na definição dos casos de teste. Os resultados obtidos são referentes à execução de cada ação de um *script*, e, também, quanto ao sucesso do teste em relação quanto à especificação declarada. O objetivo do gerenciamento dos casos de teste é a realização dos testes de regressão. Assim, é possível manter um histórico sobre a execução dos casos de testes e seus resultados durante um determinado período de tempo. O testador deverá selecionar o caso de teste e qual a ferramenta específica de funcionalidade registrada que será utilizada. A Figura 5.14 mostra a *interface* de execução dos testes de funcionalidade.

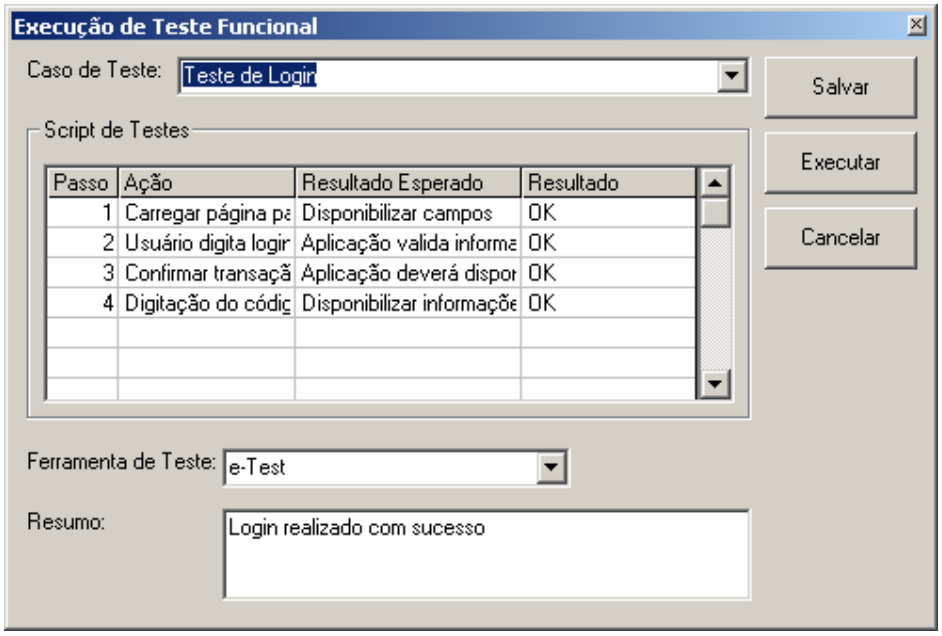


FIGURA 5.14 - Execução do Teste de Funcionalidade

Observa-se que o teste de funcionalidade abrange a verificação e validação de outros testes, como testes de links, apresentação de imagens e operações com banco de dados. Por exemplo, a maioria das ferramentas que verificam links quebrados funciona melhor com páginas estáticas que possuem uma URL para cada página. No caso de aplicações que montam páginas HTML dinamicamente, baseadas na requisição recebidas dos usuários, existe um número, potencialmente, ilimitado de páginas e são pobres candidatas para ferramentas automatizadas, isto é, não é possível empregar as ferramentas de verificação de links com sucesso.

Assim, o teste de funcionalidade tem a finalidade de verificar funcionalidades implementadas quanto à interação do usuário com a aplicação. Por exemplo, utilização de formulários, linguagens de scripts no lado cliente, interação com sistemas legados, processamento de comércio eletrônico, cookies, e outros recursos implementados característicos de aplicações baseado na Web.

Na execução do teste de navegação, deve ser informado o documento inicial que contém os links principais a serem testados. Os testes devem acontecer somente com links internos, isto é, links do domínio da aplicação. Essa configuração deve ser realizada na ferramenta a ser utilizada no momento do teste. Foram estudadas cinco ferramentas que implementam a verificação de links. Todas apresentam as mesmas características quanto à entrada da URL da página Web a ser testada e os resultados

obtidos. A Figura 5.15 apresenta a interface de entrada das informações para os testes de links.

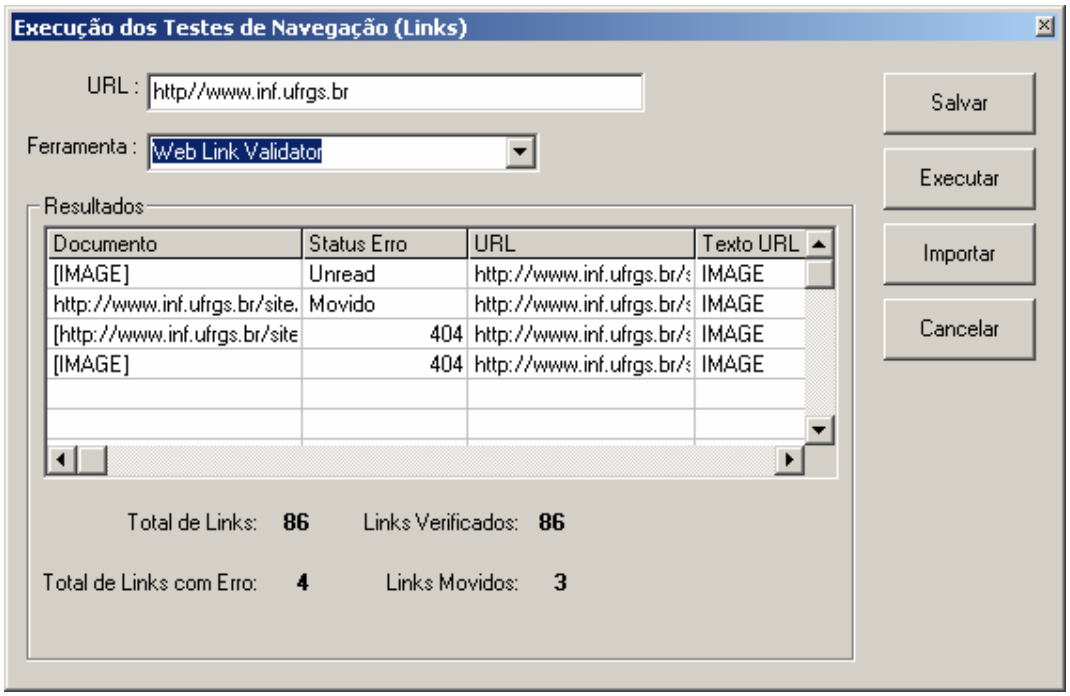


FIGURA 5.15 - Execução do Teste de Navegação (*Links*)

Após a seleção do documento inicial e escolher a ferramenta de testes de *links*, o testador deverá solicitar a execução da verificação e, após o término, importar os resultados que poderão ser visualizados como na Figura 5.15. Será possível visualizar qual o documento que contém o erro, o erro ocorrido, a URL e onde o *link* é relacionado, texto ou imagem. As estatísticas dos resultados dos testes são: o total de *links* analisados, o total de *links* com erros, *links* movidos e o total de *links* dos documentos verificados. Esses dados serão trabalhados na próxima fase, na análise de resultados.

Como nos outros tipos de testes acima, o teste de desempenho também será executado a partir das especificações nos casos de teste. Seguindo o mesmo processo, o testador deverá escolher os casos de teste e a ferramenta utilizada para o teste de desempenho. Lembrando que as especificações contidas nos casos de teste são, na realidade, a descrição de cenários de teste para as ferramentas automatizadas.

FIGURA 5.16 - Execução do Teste de Desempenho

Os resultados apresentados, após a execução dos testes, na Figura 5.16 foram levantados, como as informações mais importantes de saída no teste de desempenho. A interpretação desses resultados depende dos objetivos dos testes e de acordo com a estratégia de teste. Caso o objetivo seja o teste de carga, o qual testa a aplicação contra o número de usuários com um tempo de resposta aceitável, deverá ser dada atenção ao número de usuários virtuais nos casos de teste e o resultado do tempo de resposta obtido. Por outro lado, se o objetivo for testar a capacidade de carga da aplicação por um determinado período de tempo, então já será um teste de *Stress*. E finalmente, o teste de capacidade é usado para determinar o número máximo de usuários concorrentes que a aplicação pode gerenciar.

Os valores dos resultados apresentados foram obtidos no final da realização dos testes, tendo-se assim uma visão do comportamento da aplicação. Porém, as ferramentas possuem recursos de visualização dos testes em tempo real através de gráficos que mostram o andamento de como a aplicação está respondendo à carga submetida nos testes. Desta forma, importar os dados dos testes realizados em tempo real possui uma grande utilidade para a geração de gráficos.

Para finalizar esta seção, é interessante ressaltar que um dos objetivos principais da *WebTestManager* é justamente submeter casos de teste às ferramentas e obter os resultados após a execução. Assim, for apresentado como são as *interfaces* através dos quais o usuário irá interagir com a ferramenta. Nas seções seguintes são detalhadas as estratégias de integração.

5.3.5. Gerenciador de Ferramentas

De acordo com a arquitetura da ferramenta apresentada no início deste capítulo, o gerenciador de ferramenta tem a responsabilidade de fazer a *interface* entre o gerenciador de testes e as ferramentas específicas utilizadas na execução dos testes. Assim, a primeira atividade que deverá ser realizada será o registro de informações das ferramentas utilizadas no processo de teste. Isso permite ter um controle de utilização das ferramentas, bem como possibilitar ao testador um melhor gerenciamento das melhores funcionalidades através de um *log* de uso no decorrer do tempo.

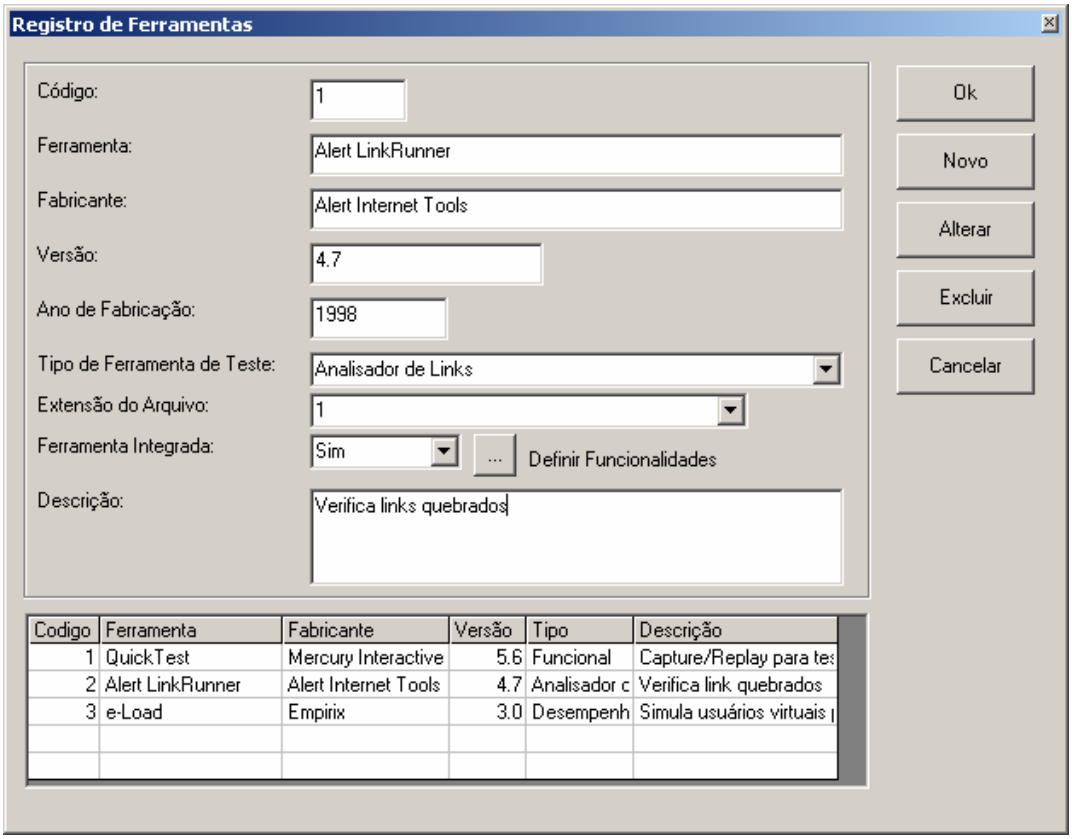


FIGURA 5.17 - Registro de Ferramentas de Teste

Como visualizado na Figura 5.17, o testador deverá informar o nome da ferramenta, fabricante, tipo, versão, ano de fabricação, extensão de arquivo utilizado de saída, uma breve descrição da ferramenta e principalmente se é integrada com a WebTestManager. Caso seja integrada, o testador deve definir as funcionalidades que podem ser compartilhadas e as informações trocadas entre as ferramentas. Esse processo de integração será comentado com mais detalhe na seção 5.4.

O gerenciador de ferramentas manterá um log de histórico de uso da ferramenta, permitindo assim a criação de um “ranking” entre os tipos de teste mais realizados.

5.3.6. Análise de Resultados

A última fase no processo é a análise de resultados obtidos com os testes. O objetivo da integração dos vários tipos de teste tem como finalidade também uma melhor visualização dos resultados gerados pelas ferramentas específicas de testes para aplicações *Web*. Manter esses resultados armazenados permite também gerar informações em relação às diversas versões dos testes, mostrando erros que a aplicação venha a apresentar após cada atualização. A seguir são mostrados os resultados que podem ser visualizados através da *WebTestManager*.

Uma forma de apresentação dos resultados é mostrá-los em relação a cada tipo de teste que foi executado e que não estariam disponíveis pelas respectivas ferramentas de teste, e que apresentam um bom instrumento para a avaliação no controle de qualidade das aplicações *Web*.

Por exemplo, no teste de *links* (navegação) são extraídas informações quanto à quantidade de *links* quebrados, movidos, órfãos, etc. Porém, as ferramentas não

gerenciam os resultados obtidos. Assim, um dos resultados que a *WebTestManger* apresenta é um gráfico indicado na Figura 5.18, que mostra o número de *links* quebrados e movidos em relação às atualizações realizadas na aplicação.

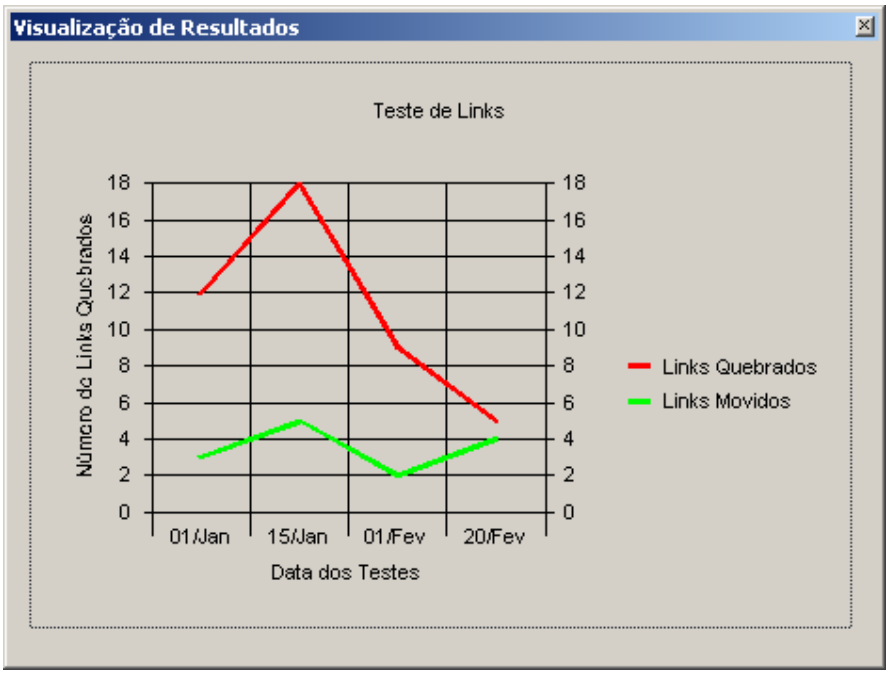


FIGURA 5.18 - Resultados do Teste de *Links*

O gráfico representa desta forma, o comportamento em relação às versões dos testes a cada atualização da aplicação. É importante deixar claro que esse resultado é direcionado para aplicações que possuem um número constante de atualização de páginas e conteúdos. Um problema encontrado é quando as aplicações geram páginas automáticas que contém *links* e que devem ser verificados também. Neste caso, o resultado dependerá da ferramenta que será utilizada para a execução dos testes.

Outros resultados que podem ser visualizados, para um mesmo tipo de teste, são em relação aos testes de desempenho de uma aplicação *Web*. Um dos resultados refere-se ao número de usuários que utilizam a aplicação e o número de ocorrências de erros com as transações solicitadas. Assim, pode-se visualizar que em um determinado momento no crescimento do número de usuários, o número de erros de algumas transações cresce também.

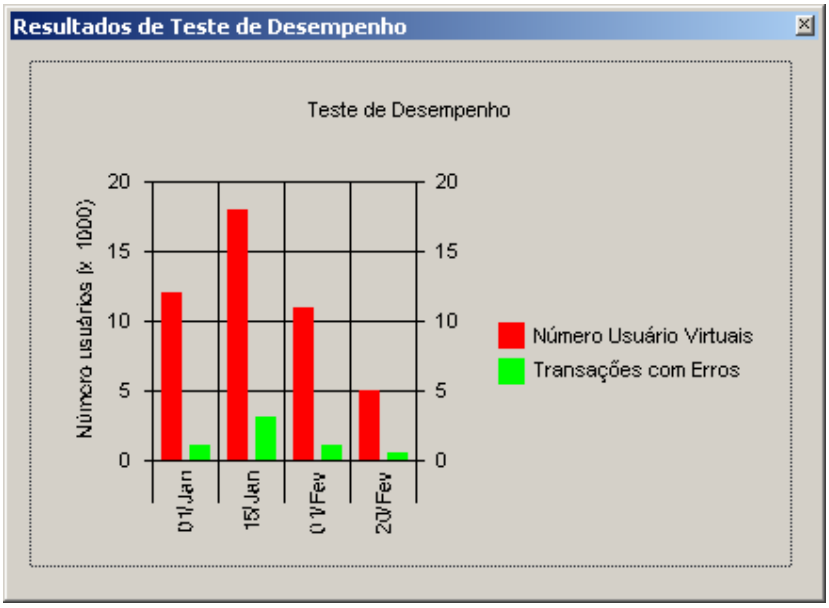


FIGURA 5.19 - Resultados dos Testes de Desepenho

A Figura 5.19 apresenta o gráfico com quatro versões do teste de desempenho com várias combinações da quantidade de usuários na aplicação (em vermelho). Para cada quantidade de usuários, é apresentado o número de transações que ocorreram com erros em unidades (em verde). Este tipo de avaliação é importante, pois mostra os riscos de possibilitar aos usuários executarem um determinado processo de negócio e ter a probabilidade de ocorrência de erros, caso tenha uma grande quantidade de usuários solicitando o serviço simultaneamente.

Outras possíveis avaliações de resultados dos testes são informações obtidas quanto aos vários tipos de testes. Por exemplo, os resultados obtidos com os testes de *links* e de desempenho podem ser visualizados quanto aos parâmetros utilizados nos dois testes, ou seja, qual seria a relação entre a quantidade de usuários utilizando a aplicação e o total de *links* quebrados. A Figura 5.20 apresenta a relação entre os resultados dos testes de desempenho e os resultados dos testes de navegação.

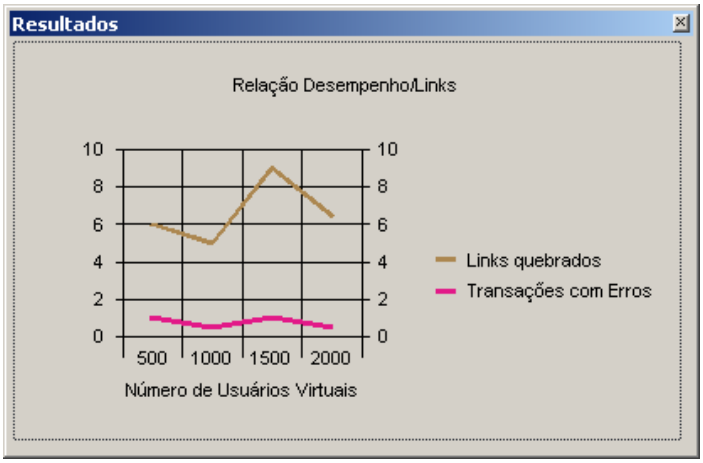


FIGURA 5.20 - Resultados do teste de desempenho versus *links*

O gráfico apresenta a visualização do resultado da relação entre os testes de desempenho e de *links*. Para cada quantidade de usuários são apresentados os números

de *links* quebrados e de erros em transações testadas no teste de desempenho. Assim, são visualizados as relações da ocorrência de *links* quebrados e os erros apresentados nas transações dos scripts de testes de desempenho.

Os outros testes abordados neste trabalho, compatibilidade e funcional, podem gerar relatórios individuais ou relacionados também. Relatórios podem ser gerados em função das informações quanto aos requisitos, casos de testes e execução dos mesmos. Outro relatório importante também é o histórico sobre a utilização das ferramentas.

Métricas de testes de aplicações *Web* podem ser visualizadas. Essas servem também de justificativa para a integração dos resultados dos testes. Patnaik (2002) define algumas métricas de teste. Uma das métricas é razão entre o número de *links* quebrados e o total de *links*. Uma aplicação não deve possuir mais do que dez por cento de *links* quebrados. Outras métricas correspondem ao teste de desempenho como: número máximo de usuários, tempo de resposta, etc. Como a *WebTestManager* trabalha com outros resultados de teste, podem-se definir outras métricas como: número de erros ocorridos em um determinado navegador ou o número de erros para uma certa função da aplicação.

5.4. Integração com outras Ferramentas

A ferramenta proposta tem como objetivo integrar diversas ferramentas de teste de aplicações baseada na *Web*, como mostrado no início do capítulo. Porém, integrar diferentes ferramentas de software de vários fornecedores requer alguns cuidados especiais. É necessário definir diversos parâmetros para troca de informações com a finalidade de executar o propósito da integração.

O objetivo maior da *WebTestManager* é importar os resultados das ferramentas de testes específicos. No entanto, em alguns casos, será necessário também informar algumas informações de configuração para a realização dos testes. Caso não seja possível a realização da integração, a ferramenta permite a entrada de dados de forma manual através de *interfaces* de entrada de dados, na qual o usuário pode informar os dados mais importantes para um determinado teste.

Para configurar uma determinada ferramenta com a finalidade de integrá-la a *WebTestManager*, é necessário especificar algumas informações de configuração relevantes para a troca de dados. Essas informações podem ser tanto dados de saída, como por exemplo, casos de teste definidos na *WebTestManager* ou informações sobre os resultados dos testes realizados, gerados pelas ferramentas a serem integradas.

A Figura 5.21 apresenta a *interface* de configuração. É necessário informar o nome da ferramenta já registrada e o local onde os arquivos de entrada e saída são armazenados. O arquivo de entrada armazena informações que vai alimentar as ferramentas de teste. Por exemplo, para a execução do teste e desempenho, é necessária a definição do cenário de teste. Assim a *WebTestManager* gera esses arquivos para serem executados pela ferramenta. Já o arquivo de saída armazena os dados dos resultados gerados após a execução do teste. Algumas ferramentas trabalham somente com a geração dos arquivos de saída.

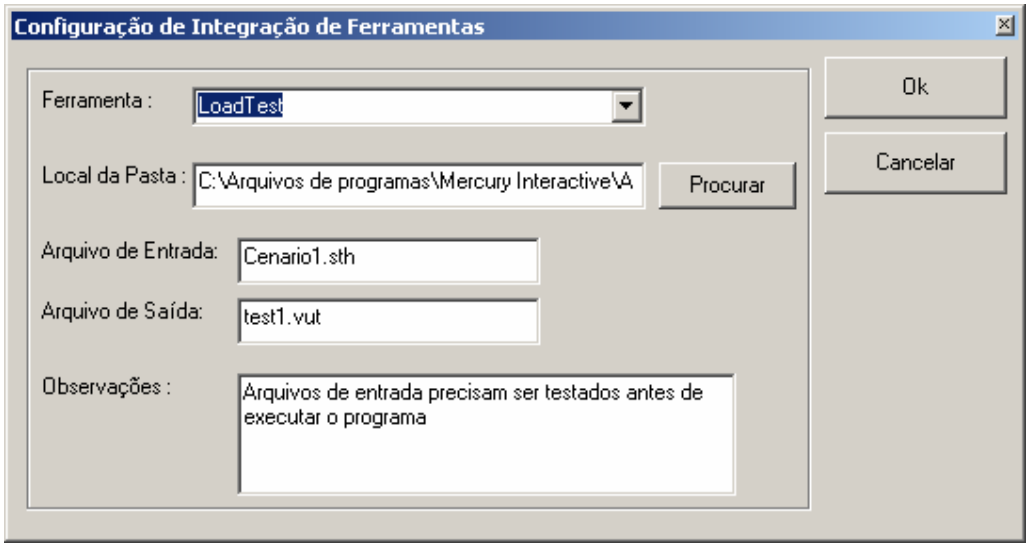


FIGURA 5.21 - Configuração de integração de ferramentas

Para cada tipo de ferramenta registrada, será necessário definir a estrutura do arquivo que a *WebTestManager* deve gerar. A seguir é apresentado um exemplo de arquivo de entrada de cenário para a ferramenta de teste de desempenho LoadTest gerado pela *WebTestManager* pela definição dos casos de teste.

```
{Product
Product=Astra LoadTest Controller
Version=7.0.0.0
}

{ScenarioGeneralConfig
ScenarioType=1
Is_mix=1
Is_runner_file=0
Runner_file=
Host_kill_type=0
GlobalDir=
}

{ScenarioPrivateConfig
Path=C:\Arquivos de programas\Mercury Interactive\Astra
LoadTest\scenario\Scenario1.lrs
Author=Administrador
Subject=Scenario1
Description=
Hosts=1
Scripts=5
Groups=4
Vusers=44
Tranzactions=0
Rendezvous=0
}
```

O exemplo acima apresenta o caso em que a *WebTestManager* gera o arquivo necessário para a execução do teste. Porém é preciso também escrever um programa para a leitura dos arquivos que contém os resultados do teste para todas as ferramentas registradas. Por exemplo, para o teste de navegação (*links*), as ferramentas geram arquivos que necessitam serem lidos pela *WebTestManager*.

5.4.1. Importação de Dados

A importação dos dados das ferramentas de teste vai depender do formato de relatórios gerados por cada ferramenta. As ferramentas geram relatórios no formato texto, outras em planilhas e outras em formato HTML. Algumas trabalham com arquivos de formato próprio ou através de vários arquivos de configuração que armazenam as informações dos resultados dos testes.

Assim, para a realização do trabalho, foram selecionadas as ferramentas em que os resultados pudessem ser importados a partir do arquivo que contém os resultados dos testes. Deste modo, foi implementado um programa para importar as informações de cada ferramenta utilizada no protótipo da ferramenta. Por isso, justifica-se a escolha da ferramenta no momento da execução dos testes. Caso uma nova ferramenta seja utilizada, é necessário desenvolver um programa para a importação das informações.

Algumas ferramentas gravam informações dos resultados do testes em formato XML, o que facilita a transferência de informações. XML é a tendência para a troca de informações entre aplicações de software, desta forma, para algumas ferramentas de teste, existe a possibilidade de integração através desta linguagem.

Por exemplo, a ferramenta **LinkAlarm** que verifica *links* quebrados gera um arquivo no formato XML com resultado da execução do teste. Desta forma, a leitura deste arquivo para a *WebTestManager* é facilitada. A seguir é apresentado um exemplo do arquivo gerado em XML.

```
<linktest>
  <linkError>
  <linkdescription>Could Not Determine Description</linkdescription>
  <url>http://www.orm.com.br/criancavia</url>
    <problem>DocumentMayHaveMovedToANewLocation</problem>
  ……
  </linkError>
 <verification Information>
      <totaltime>01 min 24 sec </Totaltime>
      <linkfoundpage>76</linkfoundpage>
      <linkchecked>76</linkchecked>
      <validlink>73</validlink>
      <invalidlink >3 </invalidlink>
      <movedlink>2 </movedlink>
  </verification Information>
</linktest>
```

É importante deixar claro que com a utilização de XML para a troca de informações dos resultados, é necessária a definição de um DTD (*Definition Type Document*) para cada ferramenta a ser utilizada, visto que como são vários fornecedores não vai haver uma padronização na geração de um formato padrão para o arquivo de resultado.

5.4.2. Vantagens e Problemas

A integração dos resultados das ferramentas apresenta inúmeras vantagens, como por exemplo: armazenamento estruturado dos resultados dos testes, visualização

temporal desses resultados, acompanhamento do andamento do processo de teste e seus resultados, reutilização dos casos de testes e geração de relatórios integrados.

Uma questão a ser considerada quanto à integração é em relação à utilização de várias ferramentas que não geram resultados padronizados. Assim, uma dificuldade seria o desenvolvimento de novos programas ou formas de leitura dos resultados para cada nova ferramenta integrada.

5.5. Conclusões

Como acontece no software convencional, o processo de teste de aplicações *Web* deve ser estruturado, organizado e gerenciado com o objetivo de encontrar o máximo de erros em um período de tempo mínimo e que seja documentado e reutilizável. A utilização de ferramentas automatizadas de teste em aplicação *Web* também se torna essencial para garantir o controle de qualidade. Alguns testes são praticamente impossíveis sem o uso de tais ferramentas.

Assim, é primordial que o processo de teste seja suportado por uma ferramenta que gerencie o planejamento, execução e resultados dos testes em aplicações *Web*, permitindo também a integração com ferramentas de testes específicos. Este capítulo apresentou a *WebTestManager*, uma ferramenta de apoio ao processo de teste de aplicações *Web* na qual foram definidos três módulos de gerenciamento: testes, ferramentas e resultados. A ferramenta permite centralizar informações dos diversos tipos de testes realizados voltados para aplicações *Web*, facilitando a avaliação dos resultados gerados.

6. Estudo de Casos

Após a ferramenta de apoio ao processo de teste de aplicações ter sido concluída. Foram escolhidas duas aplicações *Web* que possuem características que permitem utilizar os testes abordados na ferramenta, como compatibilidade, navegação, desempenho e segurança.

É importante frisar que, para os estudos de casos realizados aqui, foram aplicados estritamente testes de sistema, os quais foram ênfases neste trabalho. Assim, o estudo não foi realizado durante o processo de desenvolvimento da aplicação, mas quando da aplicação concluída, ou seja, o sobre produto.

Foram utilizadas ferramentas de teste sobre aplicações *Web* com utilização temporária, não sendo nenhuma ferramenta adquirida junto ao fornecedor.

6.1. Aplicação Bancária na Web (Home Banking)

Uma das aplicações *Web* mais utilizadas é o *Home Banking*, pois disponibiliza serviços bancários facilitando a vida dos usuários. Essa aplicação deve atender aos usuários de forma confiável e com qualidade como as versões de software dos pontos de auto-atendimento. Porém, os problemas que podem apresentar são maiores em função de serem baseados na *Web*, problemas esses discutidos no decorrer do trabalho. Para realizar este estudo de caso foi utilizado o *Home Banking* do Banco do Brasil.

Assim, testes foram planejados e executados com o objetivo de encontrar erros através dos vários tipos de testes e os resultados foram avaliados de forma a mostrar a utilidade da ferramenta desenvolvida.

Os requisitos de testes definidos para este estudo de caso foram os seguintes:

- ✓ Compatibilidade
  - Quais erros ocorrem com os casos de testes de compatibilidade definidos;
- ✓ Desempenho
  - Verificar se a aplicação suporta 1000 usuários simultaneamente verificando o extrato da conta corrente;
- ✓ Funcionais
  - Entrar na aplicação como usuário;
  - Solicitar extrato de conta corrente;
  - Fazer uma transferência entre conta corrente e poupança;
- ✓ Navegação
  - Visualizar erros nos *links*;

As execuções dos testes ocorreram durante 2 meses, e os resultados foram armazenados com a finalidade de obter como foi o andamento da aplicação diante dos testes.

6.1.1. Testes de Compatibilidade

Os testes de compatibilidade ocorreram de duas formas. A primeira foi utilizando uma ferramenta *on-line* disponível na *Web*, no endereço http://www.netmechanic.com, que verifica problemas da aplicação com os diferentes

navegadores. A segunda forma foi feita manualmente com algumas combinações de sistemas operacionais, navegadores e suas versões, e outras características para o teste de compatibilidade.

Utilizando a ferramenta disponível na URL apresentada anteriormente foram encontradas porcentagens de compatibilidade com os navegadores e algumas versões. O resultado é apresentado na Figura 6.1.

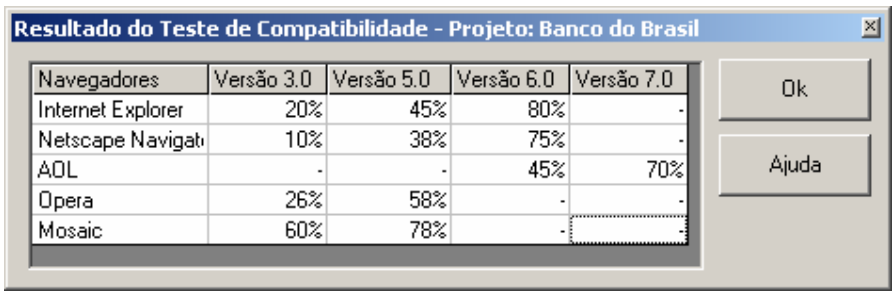


FIGURA 6.1 - Resultado do Teste de Compatibilidade (*Home Banking*)

O resultado encontrado apresenta o grau de compatibilidade somente com os navegadores e suas versões, sem levar em conta o sistema operacional. Também não são apresentados os erros que podem ocorrer quando da utilização da aplicação.

Nos testes manuais, foram definidos alguns casos de testes e executados manualmente e observados os problemas ocorridos, registrando-os na ferramenta. Os resultados estão expressos na Figura 6.2.

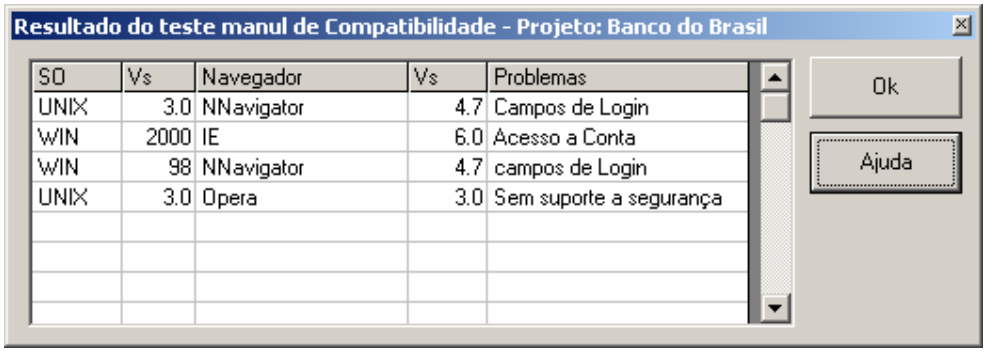


FIGURA 6.2 - Resultado do teste manual de Compatibilidade

Desta forma, verificou-se que a aplicação apresenta problemas sérios de compatibilidade em relação a algumas combinações de navegadores e sistemas operacionais. Os erros encontrados também impossibilitam os usuários a realizarem os requisitos essenciais definidos neste estudo de caso. Nos quatro testes realizados, somente o terceiro caso de teste na FIgura 6.2 não apresentou o mesmo problema.

6.1.2. Testes de Navegação e Funcionais

Os quatros testes para verificar a ocorrência de erros foram executados somente com a página principal da aplicação. Não foram executados testes com *links* de páginas após o *login* do usuário na conta bancária. Como são páginas geradas dinamicamente, as ferramentas não suportam essa verificação. Assim, só foram verificados *links* da parte

estática da aplicação. O resultado dos testes está representado na Figura 6.3, o qual indica o número de *links* quebrados para cada um dos testes.

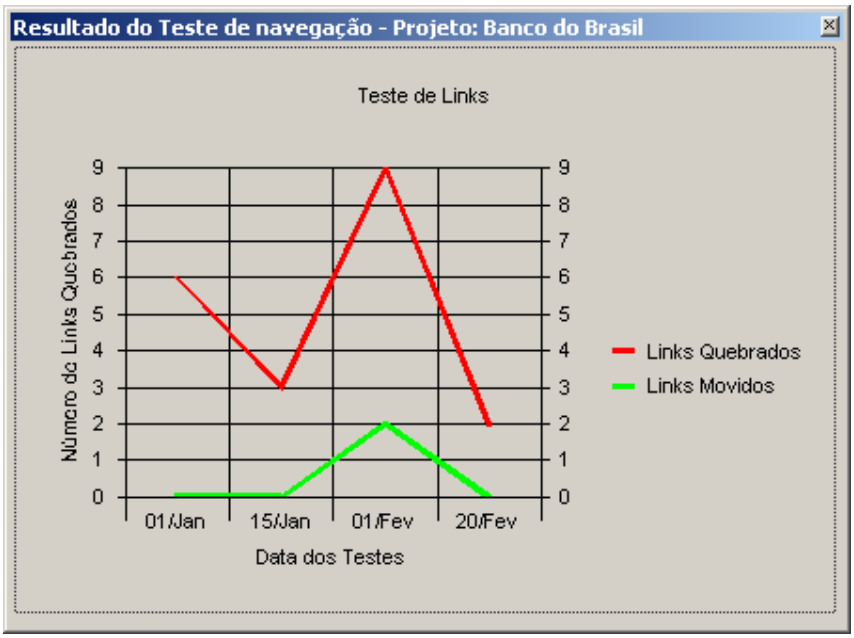


FIGURA 6.3 - Resultado do teste de *links* do Banco do Brasil

Para os testes funcionais foram gerados três *scripts* com ferramentas C/R. Em cada uma das quatro versões dos testes, foram executados os mesmos três *scripts* com a finalidade encontrar possíveis erros. A Figura 6.4 mostra os resultados em relação à quantidade de erros encontrados.

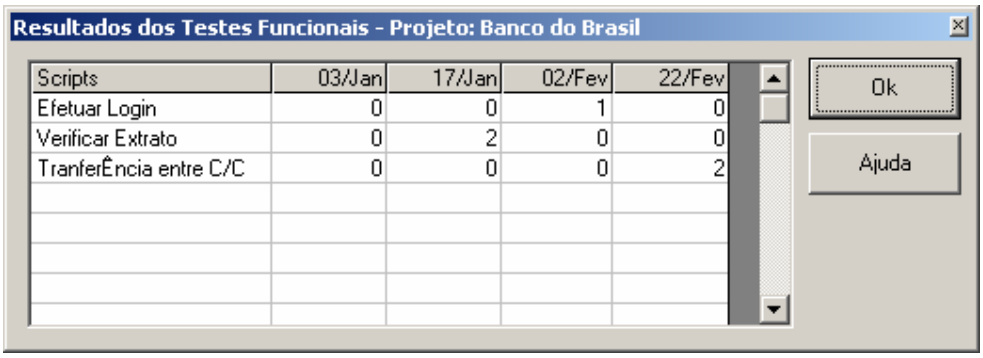


FIGURA 6.4 - Resultados dos Testes Funcionais (*Home Banking*)

As maiorias dos erros mostradas na Figura 6.4 foram decorrentes de *links* quebrados. Isso mostra que o resultado do teste de navegação pode ser obtido também através do teste funcional.

6.1.3. Testes de Desempenho

Os testes de desempenho aplicados indicaram o maior índice de erros. Foram executados 1000 usuários virtuais para cada *script* utilizado nos testes funcionais. Na medida em que o número de usuários virtuais aumentava, o número de transações com

falhas aumentava também. Nas quatro versões executadas, o comportamento em relação ao tempo de resposta não teve uma variação grande.

Os resultados encontrados expressam somente informações quanto aos processos de negócios especificados nos *scripts* de testes. Desta forma, não foi realizada uma avaliação em relação ao desempenho global da aplicação, isto é, processamento de outras transações simultaneamente.

O foco foi identificar o número de transações com erros para o determinado número de usuários virtuais fornecido. Assim, observou-se que são necessários a definição dos objetivos dos testes e os processos de negócios a serem verificados. A Figura 6.5 apresenta os resultados com os testes de desempenho para este caso de uso.

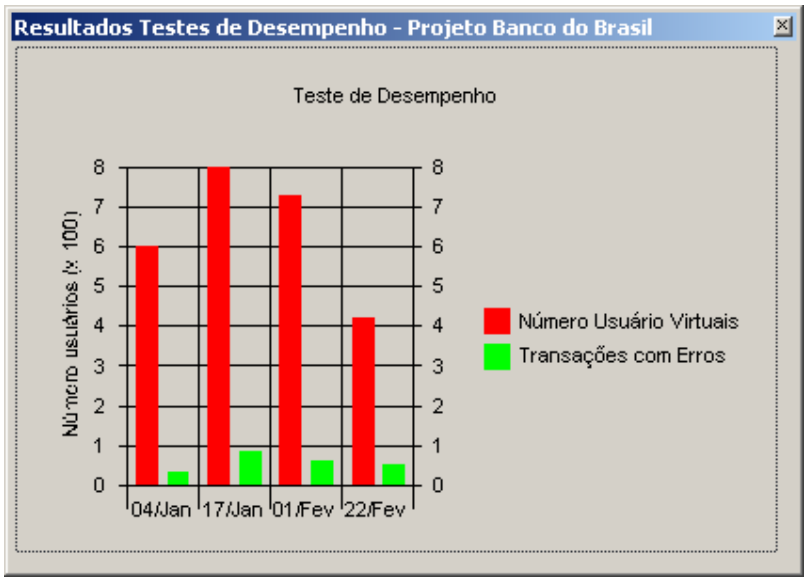


FIGURA 6.5 - Resultados dos Testes de Desempenho

Como pode ser percebido no gráfico da Figura 6.5, para as quatro versões dos testes, o número de ocorrências de erros foi maior quando o número de usuários virtuais era maior. Porém, na realização da última versão do teste, foi verificado que mesmo a queda de quase 50% no número de usuários, o número de transações com erros não teve a mesma proporção na redução de erros. Assim, percebeu-se que nem sempre a quantidade de usuários, necessariamente, indicará aumento nas transações com erros.

6.1.4. Considerações

Além da realização dos testes específicos apresentados anteriormente, nos quais foram analisados os dados de quatro versões de testes, a *WebTestManager* permitiu documentar o processo de teste. Assim, vários relatórios podem ser visualizados contendo informações dos requisitos, os respectivos casos de testes, informações das execuções e os resultados obtidos. A Figura 6.3 apresenta um exemplo de um relatório de saída com informações sobre o processo de teste realizado neste estudo de caso.

FIGURA 6.6 - Exemplo de Relatório de Planejamento do Teste

O relatório contém informações dos requisitos, casos de testes associados a estes requisitos, informações da execução, como quais as ferramentas utilizadas, e finalmente os resultados obtidos nos testes. As informações dos testes são mostradas de acordo com o tipo de teste definido. Em alguns casos, é possível visualizar os resultados obtidos de forma compartilhada, como a relação entre os testes de desempenho e *links.*

6.2. Aplicação de Reserva e Compra de Passagens Aéreas

As aplicações *Web* que permitem a realização de consulta, reserva e compra de passagens aéreas tem bastante utilização por milhares de usuários. Essas aplicações também apresentam características onde a qualidade e a confiabilidade vai determinar o sucesso ou não da aplicação. Em comparação com as aplicações de *Home Banking*, essas aplicações possuem um número menor de funcionalidades e são bem definidas como, por exemplo, somente a consulta de vôos ou a compra de bilhetes.

Este estudo de caso procurou aplicar a *WebTestManager* e ferramentas automatizadas com o objetivo de realizar os principais tipos de teste como realizado no estudo de caso anterior. Os testes também foram de compatibilidade, desempenho, navegação e funcional. A aplicação selecionada foi da companhia área GOL.

Primeiramente, o teste de compatibilidade foi realizado considerando a tabela do Anexo dois. Assim, na definição dos casos de teste, foram definidas as tecnologias que estão presentes na aplicação. As informações sobre as tecnologias foram obtidas junto à companhia. Os resultados obtidos são mostrados na Figura 6.7.

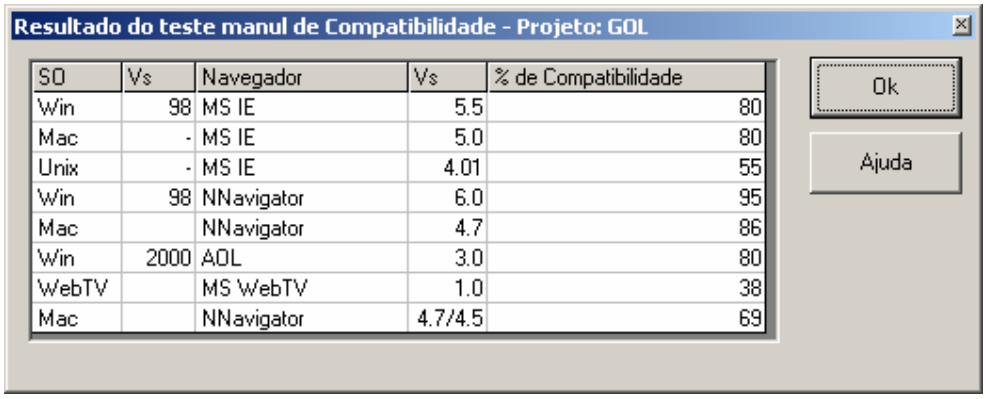


FIGURA 6.7 - Teste de Compatibilidade (*GOL Linhas Aéreas*)

As porcentagens de compatibilidade nos resultados são calculadas em função da compatibilidade de cada tecnologia empregada na aplicação. Portanto, é necessário que seja especificado pelo fornecedor do navegador o suporte para novas tecnologias. A realização do teste de compatibilidade desta forma apresenta a vantagem de verificar a compatibilidade com vários sistemas operacionais e navegadores, porém, existe a desvantagens de não possuir a visualização da execução da aplicação no respectivo sistema operacional e navegador. Isso pode ser prejudicial, pois nem todos os problemas podem ser detectados somente com informações sobre a aplicação.

No teste funcional os requisitos e casos de teste foram voltados para a compra de bilhetes, ou seja, se todas as funções implementadas na aplicação quanto o processo de compra de bilhetes foram executadas com sucesso. Assim, foram identificados os principais requisitos para a compra do bilhete como:

- Seleção do trecho, data da viagem e informações dos passageiros;
- Seleção do vôo disponível;
- Cadastro de informações do passageiro;
- Confirmação de compra através de informação do número de cartão de crédito.

Portanto, os casos de teste foram especificados com as operações citadas acima gerando assim os *scripts* para que sejam executados nas ferramentas automatizadas. Como o estudo de caso foi realizado utilizando a aplicação já disponível na *Web* não foi possível completar a compra através da informação com o número de cartão de crédito. Os *scripts* gerados atendem parte da compra, já os *scripts* de consultas de vôo foram realizados normalmente. Os testes foram executados em quatro versões, para cada versão são apresentados o número de erros encontrados, como mostra a Figura 6.8.

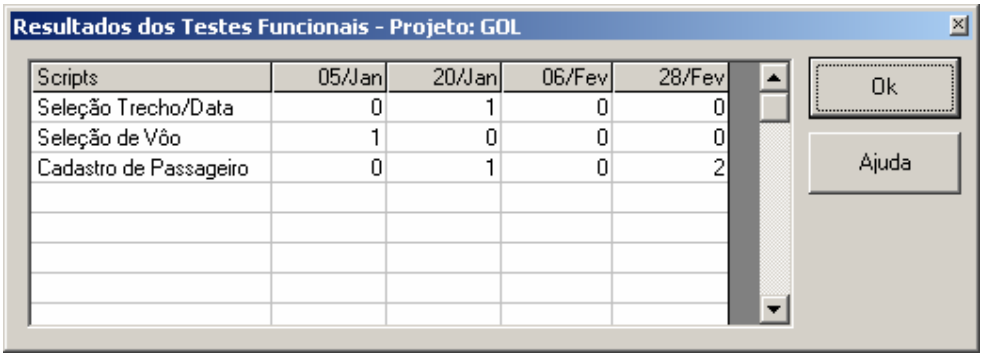


FIGURA 6.8 - Teste Funcional (*GOL Linhas Aéreas*)

Os resultados obtidos mostram que os maiores erros ocorreram em relação ao cadastro de passageiros. É importante abordar que os erros encontrados às vezes não são erros de desenvolvimento da aplicação. Pode ser um problema decorrente das outras camadas da aplicação Web. Como Nguyen (2001) cita que erros em aplicações Web podem ser sintomas de outros problemas.

Os testes de navegação não foram realizados em função das ligações entre páginas que são realizadas de forma dinâmica, pois as ferramentas que verificam os links não oferecem suporte. Assim, a navegação fica sendo testada dentro da execução do teste funcional. As ferramentas também ficaram limitadas para verificar os links existentes na aplicação, em função da tecnologia empregada.

O teste de desempenho para esta aplicação tem como principal finalidade verificar o quanto a aplicação suporta de quantidade de usuários executando o processo de compra de passagens com um tempo de resposta aceitável. Assim, foram gerados casos de teste especificando uma quantidade de dez mil usuários virtuais executando o processo de compra de bilhetes, também em quatro versões. Os scripts utilizados dos usuários virtuais são os definidos para o teste funcional. A Figura 6.9 apresenta os resultados gerados.

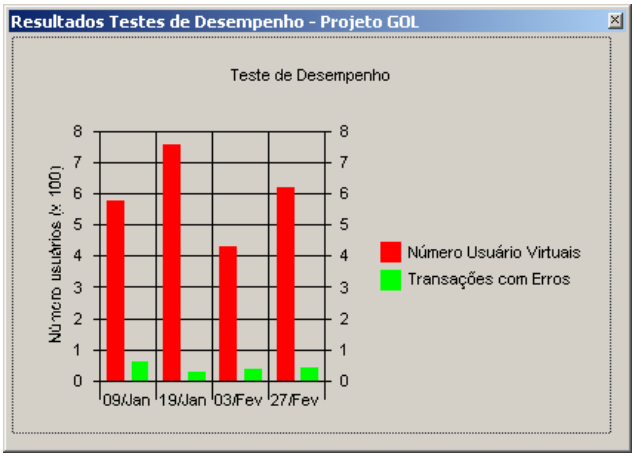


FIGURA 6.9 - Teste de Desepenho (*GOL Linhas Aéreas*)

6.3. Conclusões

Os estudos de casos realizados neste trabalho apresentaram os resultados dos testes executados e que foram planejados pela WebTestManager. As aplicações selecionadas apresentam todos as características para a realização dos testes abordados

no trabalho. Algumas dificuldades foram encontradas na realização do teste de navegação em função das limitações das ferramentas disponíveis. Como os teste foram executados com a aplicação já disponibilizada, algumas funções não foram testadas. Mas o objetivo foi alcançado quanto ao planejamento e integração com as ferramentas automatizadas.

Um importante recurso implementado pela WebTestManager utilizado nos estudos de caso foi o armazenamento dos casos de teste. Foi possível a utilização novamente dos casos de teste para as novas versões do teste. A geração dos resultados de forma organizada e em formato de gráficos também facilitou a visualização dos resultados dos diversos tipos de teste.

7. Conclusão

Neste trabalho foi apresentada uma proposta de ferramenta de apoio ao processo de teste de aplicações *Web*. A abordagem principal foi o planejamento do teste e o gerenciamento de ferramentas automatizadas. O objetivo foi de centralizar os resultados de diversos tipos de teste com a finalidade de obter informações sobre o maior número de erros possível cometidos durante o desenvolvimento das aplicações testadas.

A fase de teste faz parte da nova disciplina que surgiu com o crescimento de soluções de software para *Web*, chamada de Engenharia de Software para a *Web* ou Engenharia da *Web*. O propósito da fase de teste é a garantia de qualidade, visto a complexidade que envolve o desenvolvimento de aplicações de software baseado na *Web*. Portanto, o teste deve estar presente em todo projeto que exija o mínimo de qualidade como em todo projeto de software tradicional.

Os desafios da atividade de teste são as características especiais presente na arquitetura de aplicações *web* através de diversas tecnologias de *hardware* e *software*, tais como a utilização de redes, servidores, roteadores, sistemas operacionais e navegadores, entre outros. Conseqüentemente, o teste de aplicações *Web* deve ter uma atenção especial por parte da equipe de desenvolvimento.

Para a realização deste trabalho, foi feito um levantamento dos principais tipos de teste que devem ser executados sobre aplicações *Web*. Vários autores apresentam abordagens das atividades de teste e suas técnicas, porém existe um consenso sobre os testes que devem ser executados. Um dos principais tipos de teste é o de navegação, cujo objetivo é encontrar erros de ligação entre as páginas *web*. A maior incidência de erros com aplicações *Web* são os *links* quebrados, ou seja, a impossibilidade de visualizar outras páginas contendo informações ou serviços. Outro importante teste é o de compatibilidade cuja finalidade é verificar como a aplicação se comporta diante a combinação de plataformas de *hardware* e *software*. Assim problemas de erros de compatibilidade inviabilizam a execução de várias funções da aplicação. Essas funções também podem ser testadas através dos testes funcionais, que têm a intenção de verificar se funções específicas da aplicação desenvolvidas estão executando de forma satisfatória. E por fim, um teste bastante requisitado é o de desempenho, cujo propósito é conferir se a aplicação apresenta erros quando um determinado número de usuários acessa simultaneamente os serviços da aplicação.

Outro importante aspecto quando da realização do teste é o modelo de processo a ser seguido pelo testador ou equipe de teste, isto é, quais as atividades devem ser realizadas para a execução do teste de forma organizada e documentada. O processo de teste abordado neste trabalho teve o foco na definição de requisitos, no planejamento de casos de teste, na execução do teste e na análise dos resultados obtidos. Um importante recurso da implementação neste processo de teste é o gerenciamento das informações dos casos de teste e dos resultados gerados pelas ferramentas de teste.

O projeto e implementação da *WebTestManager* possibilitaram o acompanhamento de todas as fases no processo de teste e também o gerenciamento da utilização das ferramentas automatizadas referentes aos tipos de teste de aplicações baseadas na *Web*. Com a integração com esses produtos de software comerciais, buscou-se facilitar a execução automatizada do teste, visto que para alguns tipos de teste a execução manual é de difícil realização. Outra facilidade é o gerenciamento dos resultados do teste gerados por essas ferramentas para cada versão do teste.

A ferramenta foi estruturada em três módulos principais com finalidades específicas, porém integradas: planejamento, execução e resultados. A definição dos modelos de classes da ferramenta permitiu a visualização desses módulos com os seus respectivos objetos de trabalho.

Assim, a modelagem da *WebTestManager* apresenta uma estrutura, que facilite que novos tipos de teste sejam incorporados através de novas classes de teste e também que novas ferramentas de teste comerciais sejam integradas para a execução do teste. O propósito foi definir um ambiente no qual todos os tipos de teste possam ser planejados e gerenciados.

Uma funcionalidade projetada para uso na *WebTestManager*, e que não foi identificada em outras ferramentas de planejamento de teste para aplicações *Web*, é o controle de qualidade da aplicação quanto às métricas de teste. É possível acompanhar, através dos resultados do teste, se uma determinada aplicação se encontra dentro dos padrões de qualidade.

Para implementação do protótipo da *WebTestManager* procurou-se dar ênfase ao apoio do processo, através do planejamento das fases de teste e a integração com algumas ferramentas de teste automatizadas. Desta forma, não foram implementadas formas de planejar e executar alguns tipos de teste, como por exemplo, o teste de segurança.

Duas aplicações de software baseadas na *Web* foram submetidas a *WebTestManager*. São aplicações que representam bem todas as questões relativas quanto à necessidade do teste, tanto no aspecto de planejamento como na execução automatizada. Todos os testes realizados foram testes de sistemas, cujo objetivo foi descobrir erros no produto final da aplicação, assim não foram considerados no planejamento os testes de unidade e integração, uma vez que, aplicações *web* apresentam características de funcionamento somente quando seus componentes são executados simultaneamente.

Os resultados obtidos com a implementação dos estudos de caso mostraram a quantidade de informações que devem ser gerenciadas no planejamento do teste, como os casos de teste para todos os tipos de teste e os dados dos resultados gerados pelas ferramentas após a execução dos testes. Portanto, identificou-se a necessidade de acompanhamento e gerenciamento das informações de teste com a finalidade de auxiliar o controle da qualidade da aplicação através da atividade de teste.

7.1. Contribuições

Esta dissertação propôs uma ferramenta para apoiar o processo de aplicações *Web* e o gerenciamento de ferramentas automatizadas na execução do teste. Assim, foi realizado um levantamento dos principais testes aplicados a aplicações *Web*. Para cada tipo de teste, foram apresentadas sua particularidade e a importância para realizar tal teste. O relacionamento entre os principais tipos de teste também foi apresentado, mostrado assim como as diversas áreas de teste possuem algum tipo de ligação entre si.

A *WebTestManager* incrementou a implementação na realização de teste de aplicações *Web*, propondo uma forma de trabalhar com todos os resultados dos testes que devem ser realizados com aplicações baseadas na *Web*. A ferramenta definiu um processo de teste a ser seguido de forma automatizada como forma de facilitar o planejamento do teste. Para tanto, foram implementadas *interfaces* para definição de casos de teste para cada tipo de teste, assim como, o gerenciamento da execução.

E como principal enfoque, foi implementado um ambiente para a realização de testes de aplicações *Web* através da integração de ferramentas automatizadas para os principais tipos de teste que devem ser executados. Desta forma, procurou-se estender o planejamento e execução dos tipos de testes não suportados por ferramentas de planejamento de teste existentes no mercado com características voltadas para aplicações *Web*.

7.2. Trabalhos Futuros

Como trabalhos futuros para melhoramentos da WebTestManager, pode-se citar:

- A extensão para outros tipos de teste, como teste de segurança. O protótipo da ferramenta abordou cinco tipos principais de acordo com a classificação de Ramler (2002). Outros tipos de teste e as ferramentas associadas não foram estudados no trabalho;

- Módulo para indicar possíveis soluções para os erros encontrados no teste. Desta forma, o testador já teria uma lista de sugestões para um determinado erro, diminuindo o tempo de manutenção da aplicação;

- Programação da realização do teste, de modo que seja possível programar a execução do teste de acordo com a necessidade do testador. A vantagem seria escalonar a execução do teste entre um período de tempo, possibilitando o monitoramento da aplicação.

Referências

AMBLER, S.W. Lessons in Agility from Internet-Based Development. **IEEE Software**, Los Alamitos, v. 19, n. 2, p. 66-73, Mar./Apr. 2002.

ATLANTIS, S. P. A. et al. **Ubiquitous Web Applications.** New York: Prentice Hall, 2001.Relatório Técnico.

ATZENI, P.; MERIALDO, P.; SINDONI, G. Web Site Evaluation: methodology and case study. In: INTERNATIONAL WORKSHOP ON DATA SEMANTICS IN WEB INFORMATION SYSTEMS, DASWIS, 2001, Yokohama. **Proceedings...** [S.l.: s.n.], 2001**.**

BINDER, R.V. **Testing Object-Oriented Systems:** models, patterns and tools. New York: Addison-Wesley, 1999.

BOEHM, B.; IN, H. Identifying Quality-Requirement Conflicts. **IEEE Software**, Los Alamitos, v. 13, n. 2, p. 25-35, Mar. 1996.

COLLINS, R. W. Software Localization for Internet Software: Issues and Methods. **IEEE Software**, Los Alamitos, v. 19, n. 2, p. 74-80, Mar./Apr. 2002.

DESHPANDE, Y.; HANSEN, S. Web Engineering: creating a discipline among Disciplines**. IEEE Multimedia**, Los Alamitos, v. 8, n. 2, p. 82-87, Apr./Sept. 2001.

DUSTIN, E.; RASHKA, J.; McDIARMID, D. **Quality Web Systems.** Boston: Addison Wesley, 2001.

FEWSTER, M.; GRAHAM, D. **Software Test Automation Effective Use of Test Execution Tools.** New York: Addison-Wesley, 1999.

GERRARD, P. **Risk-Based E-Business Testing, Part 1:** risks and test strategy. White Paper, System Evoluif, 2000. Disponível em: <http://www.evolutif.co.uk/articles/EBTestingPart1.pdf >. Acesso em: 17 jul. 2001.

GINIGE, A.; MURUGENSAN, S. Web Engineering: an Introduction**. IEEE Multimedia**, Los Alamitos, v. 8, n. 1, p. 14-18, Jan./Mar. 2001.

GLASS, R.L. Has Web Development Changed the Meaning of Testing? In: INTERNATIONAL WORKSHOP ON WEB SITE EVOLUTION, 2000. Proceedings... Disponível em: <http://www.stickyminds.com/pop_print.asp?ObjectId=2163&ObjectType>. Acesso em: 10 nov. 2001.

GRAHAM, D. Requirements and Testing: seven missing-link myths. **IEEE Software**. Los Alamitos, v.19, n. 5, p. 15-17, Sept./Oct. 2002.

HELM, J. C. Web-Based Application Quality Assurance Testing. In: INTERNATIONAL WORKSHOP ON WEB SITE EVOLUTION, 2001. Montreal.

Proceedings...[p. 43-52], 2001.

HETZEL**, W. Guia Completo ao Teste de Software.** Rio de Janeiro: Campus, 1987.

HIEALT, E.; MEE, R. Going Faster: testing the Web application. **IEEE Software**, Los Alamitos, v. 19, n. 2, p. 60-65, Mar./Apr. 2002.

HO, J. Evaluating the World Wide Web: a global study of commercial sites. **Journal of Computer-Mediated Communication**, Toronto, v. 3, n. 1, June 1997. Disponível em: <http://www.ascusc.org/jcmc/issue1/ho.html>. Acesso em: out. 2001.

HOM, J. **The Usability Methods Toolbox.** 1998. Disponível em: <http://www.best.com/~jthom/usability/>. Acesso em: 13 set. 2001.

HOWER, R. Beyond Broken Links. **DB Magazine,** 2001. Disponível em: <http://www.dbmsmag.com/9707i03.html>. Acesso em: 05 ago. 2001.

HOWER, R. **Web Site Test Tools and Site Management Tools**. Toronto: Software QA and Testing Resource Center, 1997. Disponível em: <http://www.softwareqatest.com/qatweb1.html>. Acesso em: 12 jun. 2001.

JACOBSON, I.; BOOCH, J.; RUMBAUGH, T. **The Unified Software Development Process**. Boston: Addison Wesley, 1999.

KALLEPALLI, C. Meansuring and Modeling Usage and Reliability for Statistical Web Testing. **IEEE Trasactions on Software Engineering,** New York, v. 27, n. 11, p. 45-55, Nov. 2001.

KALLEPALLI, C.; TIAN, J. Usage Measurement for Statistical Web Testing and Reliability Analysis. **IEEE Trasactions on Software Engineering,** New York, v. 27, n. 11. p. 23-32, Nov. 2001.

KANER, C.; FALK, J.; NGUYEN, H. Q. **Testing Computer Software.** New York: John Wiley & Sons, 1999.

KAUFMAN, E. **Testing Your Web site.** Testers' Network, VeriTest 1999. Disponível em: <http://www.veritest.com/testers'network/Web_testing1-1.asp>. Acesso em: 06 out. 2001.

KUNG, D.C.; LIU, C.H.; HSIA, P. Object-Based Data Flow Testing of Web Applications. **IEEE Software,** Los Alamitos, v. 5, n. 2, p. 7-19, Sept./Oct. 2000.

KUNG, D.C.; LIU, C.H.; HSIA, P. Structural Testing of Web Applications**. IEEE Software**. Los Alamitos, v. 5, n. 2, p. 84-99, Jul. Aug. 2000.

LINZ, T.; DAIGL, M. **How to Automate Testing of Graphical User Interfaces.** New York: Imbus GmbH, 1998. Disponível em:

<http://www.imbus.de/forschung/pie24306/gui/aquis-full_paper-1.3.html>. Acesso em: 17 set. 2001.

LYNCH, P.J.; HORTON, S. **Web Style Guide**: basic design principles for creating Web Sites. Yale: Yale University, 1999. Disponível em: <http://info.med.yale.edu/caim/manual/index.html>. Acesso em: 11 nov. 2001.

MACINTOSH, A.; STRIGEL, W. **The Living Creature – Testing Web Applications.** Quality Week 2000, San Francisco. Disponível em: <http://www.qalabs.com/resources/thelivingcreature.pdf>. Acesso em: 13 out. 2001.

MARGARIA, T.; NIESE, O.; STEFFEN, B. Demonstration of an Automated Integrated Test Environment for Web-based Applications. In: INTERNATIONAL SPIN WORKSHOP ON MODEL CHECKING OF SOFTWARE, SPIN, 9., 2002. **Proceedings...** [S.l.: s.n.] 2002. p. 250-253. Disponível em: <http://ls5-www.cs.uni-dortmund.de/~niese/Niese02c.pdf>, Acesso em: 20 mar. 2002.

MARICK, B. **Maybe Testers Schouldn't Be Involved Quite So Early**. Quality Week 2000. San Francisco. Disponível em: <http://www.testing.com/writings/testers-upstream.html>. Acesso em: 20 set. 2001.

McCALL, J.A., RICHARDS, P.K., WALTERS, G.F. **Factors in Software Quality,** [S.l.], v.3, n.1, p .77-369, 1977.

MILLER, E. **The WebSite Quality Challenge**. 2000. Disponível em: <http://www.soft.com/eValid/Technology/White.Papers/website.quality.challenge.html>. Acesso em: 10 set. 2001.

MILLER, E. **WebSite Loading and Capacity Analysis**. 2001. Disponível em: <http://www.soft.com/eValid/Technology/White.Papers/website.loading.challenge.html>. Acesso em: 09 set. 2001.

MURUGESAN, S. et al. Web Engineering: a new discipline for Web-Based System Development. In: WORKSHOP ON WEB ENGINEERING. 1999, **Proceedings…**[S.l.:s.n.], 2001.

MYERS, G. T. **The Art of Software Testing.** New York: John Wiley & Sons, 1979.

NGUYEN, H. Q. **Testing Applications on the Web.** New York: John Wiley & Sons, 2001.

NIELSEN, J. **Accessible Design for Users with Disabilities.** Alertbox 1996. Disponível em: <http://www.useit.com/alertbox/9610.html>. Acesso em: 02 mai. 2001.

NIELSEN, J.; NORMAN, D.A. **Usability On The Web Isn't A Luxury.** InformationWeek 2000. Disponível em:

<http://www.informationweek.com/773/web.htm>. Acesso em: 09 dez. 2001.

NIESE, O.; MARGARIA, T.; STEFFEN, B. Automated Functional Testing of Web-based Applications. In: INTERNATIONAL CONFERENCE ON SOFTWARE AND INTERNET QUALITY WEEK EUROPE, QWE, 5., 2002. **Proceedings...**[S.l.:s.n.], 2002. Disponível em: <http://ls5-www.cs.uni-dortmund.de/~niese/Niese01d.pdf>, Acesso em: 20 mar. 2002.

OCAMPO, G. **Testing Considerations for Web-Enabled Applications.** Testers Network, VeriTest, 1999. Disponível em: <http://www.veritest.com/Testers'Network/testing_considerations1.asp>. Acesso em: 23 ago. 2001.

OFFUT, J. Quality Attributes of Web Software Applications. **IEEE Software**, Los Alamitos, v. 19, n. 2, p. 25-32, Mar./Apr. 2002.

OLSINA, L.; GODOY, D.; LAFUENTE, G.J.; ROSSI, G. Specifying Quality Characteristics and Attributes for Websites. WEB ENGINEERING WORKSHOP, WWW8, 1999, Toronto. **Proceedings...** Disponível em: <http://budhi.uow.edu.au/web-engineering99/accepted_papers/olsina.pdf>. Acesso em: 21 out. 2001.

OLSINA, L.; GODOY, D.; LAFUENTE, G.J.; ROSSI, G.: Quality Characteristics and Attributes for Academic Web Sites. WEB ENGINEERING WORKSHOP, WWW8, 1999, Toronto. **Proceedings...** Disponível em: <http://budhi.uow.edu.au/web-engineering99/accepted_papers/olsina.pdf>. Acesso em: 20 out. 2001.

PATNAIK, S.; SRINIVAS, J. An Approach to Formulating Software Test Metrics for Web based Applications. In: INTERNATIONAL WORKSHOP ON WEB SITE EVOLUTION, WSE, 2000, Zurich. **Proccedings...** [S.l.:s.n.], 2000.

PERRY, W. E. **Effective Methods for Software Testing.** New York: John Wiley & Sons, 2000.

POMBERGER, G.; BLASCHEK, G. **Object Orientation and Prototyping in Software Engineering.** New York: Prentice Hall, 1996.

POWER, T. **Web Site Engineering.** New Jersey: Prentice-Hall, 1998.

PRESSAM, R. **Engenharia de Software.** 4.ed. São Paulo: Makron Books do Brasil, 1995.

PRESSAM, R. **Engenharia de Software.** 5.ed. São Paulo: Makron Books do Brasil, 2000.

RAMLER, R.; SCHWINGER, W.; ALTMANN, J. **Testing Web Quality Aspects.** In: IBERO AMERICAN CONFERENCE ON WEB ENGINEERING, ICWE, 2002, Santa Fé. **Proccedings...** [S.l.:s.n.], 2002.

RICCA, F.; TONELLA, P. Analysis and Testing of Web Application. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 2001, Toronto. **Proccedings...** [S.l.:s.n.], 2001.

RICCA, F.; TONELLA, P. Building a Tool for the Analysis and Testing of Web Applications: problems and solutions. In: INTERNATIONAL CONFERENCE AND TOOLS AND ALGORITHMS FOR THE CONSTRUCTION AND ANALYSIS OF SYSTEMS, TACAS, 2001, Genova, Italy. **Tools and Algorithms for the Construction and Analysis of System: Proceedings...** Berlin: Springer-Verlag, 1999. p. 373-388.

RICCA, F., TONELLA, P. Web Testing Process. **Annals of Software Engineering**, Roma, v. 14, p. 93-114, 2002.

RICE, R.W. **How to Develop Test cases and Test Scripts fo Web Application.** Disponível em: <http://www.riceconsulting.com/web_test_cases.htm>. Acesso em: 27 jun. 2001.

ROBISON, R. **Automation Test Tools Comparison.** 2001. Disponível em: <http://www.stickyminds.com/docs_index/XUS690378file1.doc>. Acesso em: 02 out. 2001.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K.C. **Qualidade de Software:** Teoria e Prática. São Paulo: Prentice Hall, 2001.

RYDÉN, J.; SVENSSON, P. **Web Application Testing**: a methodology with Tools and Considerations. 2001. Dissertação (Mestrado) - Department of Transportation and Logistics, Göteborg.

SAVOIA. A. **The Science of Web Site Load Testing**. Disponível em: <http://www.stickyminds.com/docs_index/XDD1939filelistfilename1.pdf>. Acesso em: 10 mai. 2001.

SHAH, B. Validation and Verification Testing of e-Commerce Applications. In: INTERNATIONAL SOFTWARE TESTING CONFERENCE, 2001, Boston. **Proceedings...** Disponível em: <http://www.softwaredioxide.com/Channels/Content/Shah_calidationand_verification_of_testing_ecommerce_application.pdf>. Acesso em: 17 out. 2001.

SIEGEL, D. **Creating Killer Web Sites.** 2nd ed. New York: Hayden Books, 1997.

SOBERANO, Vicent. Testing Web Application. In: INTERNATIONAL CONFERENCE QUALITY SOFTWARE, 2000. Disponível em: <http://members.spree.com/oceansurfer/webtesting.htm>. Acesso em: 15 abr. 2001.

SOMMERVILLE, I. **Software Engineering**. 6th ed. New York: Addison-Wesley, 2001.

SPLAINE, S.; JASKIEL, S. P. **The Web Testing Handbook.** Orange Park: STQE Publishing, 2001.

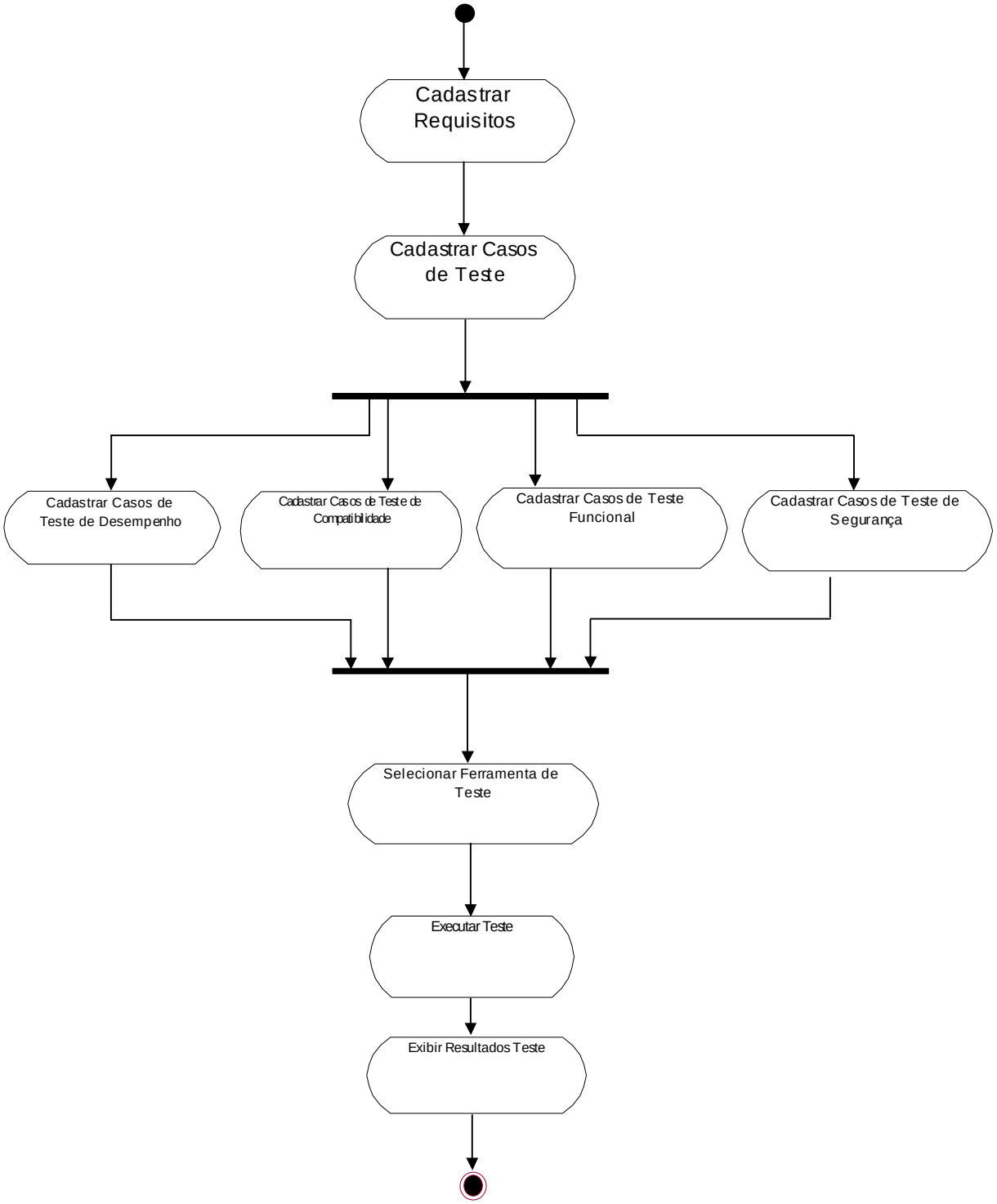
STOTTLEMYER, D. **Automated Web Testing Toolkit.** New York: John Wiley & Sons, 2001.

STOUT, G.A. **Testing a Website**: best practices. Berlim: The Revere Group, 2001. Disponível em: <http://www.stickyminds.com/docs_index/XUS893545file1.pdf>. Acesso em: 13 jan. 2002.

TONGEREN, T.V. **Compatibility and Security Testing of Web based Application.** Quality Week, 1999. Disponível em: <http://www.soft.com/News/TTN-OnLine/ttnmar99.html>, Acesso em: 12 set. 2001.

WU, Y. Testing Web based Application. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 2001, Ontario. **Proceedings...** [S.l.: s.n.], 2001. p. 25-34. Disponível em: <http://www.ise.gmu.edu/~wuye>. Acesso em: 16 jun. 2001.

Anexo 1 Diagrama de Atividades



Anexo 2 Tabela de Compatibilidade

SO	Navegador	Java	frames	tables	plug-ins	jscript	CSS	gif89	dhtml	I-frames	XML
Win	MS IE 5.5	S	S	S	S	S	S	S	S	S	P
Win	MS IE 5.0	S	S	S	S	S	S	S	S	S	p
Win	MS IE 4.0	S	S	S	S	S	S	S	S	S	n
Win	MS IE 3.0	S	S	S	S	S	P	S	N	S	N
Win	MS IE 2.0	N	N	S	N	N	N	N	N	N	N
Mac	MS IE 5.0	S	S	S	S	S	P	S	S	S	P
Mac	MS IE 4.0	S	S	S	S	P	P	S	S	S	N
Mac	MS IE 3.0	S	S	S	S	P	P	S	N	S	N
Mac	MS IE 2.0	N	S	S	S	N	N	N	N	N	N
UNIX	MS IE 4.01	S	S	S	S	P	P	S	S	S	N
Win	NN 6	S	S	S	S	S	P	S	S	S	P
Win	NN 4.7/4.5	P	S	S	S	P	P	S	S	N	N
Win	NN 4	S	S	S	S	P	P	S	S	N	N
Win	NN 3.0	S	S	S	S	P	N	S	N	N	N
Win	NN 2.0	S	S	S	S	P	N	S	N	N	N
Mac	NN 4.7/4.5	N	S	S	S	P	N	S	S	N	N
Mac	NN 4.06	S	S	S	S	P	N	S	S	N	N
Mac	NN 3.0	S	S	S	S	P	P	S	N	N	N
Mac	NN 2.0	N	S	S	S	P	N	S	N	N	N
UNIX	NN 4.06	S	S	S	S	P	N	S	S	N	N
UNIX	NN 3.0	S	S	S	S	P	N	S	N	N	N
UNIX	NN 2.0	S	S	S	S	P	N	S	N	N	N
OS/2	NN 2.02	N	N	S	N	N	N	S	N	N	N
Win	Opera 4.02	P	S	S	S	P	P	S	P	S	P
Mac	Mosaic 3.07	N	S	S	N	N	N	N	N	N	N
Win	Mosaic 3.0	N	S	S	N	N	N	N	N	N	N
Win	AOL 3.0	N	S	S	N	N	N	N	N	N	N
Mac	AOL 2.7	N	N	N	N	N	N	N	N	N	N
Win	AOL 1.0	N	N	N	N	N	N	N	N	N	N
Mac	AOL 1.0	N	N	N	N	N	N	N	N	N	N
Win	Lynx	N	S	S	N	N	N	N	N	N	N
UNIX	Lynx	N	S	S	N	N	N	N	N	N	N
OS/2	Lynx	N	S	S	N	N	N	N	N	N	N
NextStep	OmniWeb 2.1	N	S	S	N	S	N	N	S	N	N
NextStep	OmniWeb 1.0	N	N	N	N	N	N	N	N	N	N
WebTV	MS WebTV	N	S	S	N	P	N	S	N	N	N

Legenda:

S – suportado N – não suportado P – parcialmente suportado

Fonte :

http://www.webreview.com/browsers/browsers.shtml

Anexo 3 Dicionário de Dados

Classe : *Projetos*

Atributo	**Descrição**	**Tipo**
# Cd_Projeto	Código do Projeto	Texto
Nome	Nome do Projeto a ser testado	Texto
DataInclusão	Data da inclusão do projeto da ferramenta	Data
DataAtualização	Data da última atualização do projeto	Data
Status	Mostra a situação de andamento de teste	Texto
TipodeProjeto	Indica qual o tipo de projeto de aplicação testada	Texto
Duração	Tempo estimado do projeto	Texto

Classe: Requisitos

Atributo	**Descrição**	**Tipo**
# Cd_Requisito	Código do Requisito de Teste	Texto
Nome	Nome requisito de teste	Texto
Descrição	Informa de forma detalhada o objetivo do requisito	Texto
Tipo	Tipo do teste de Requisito	Texto
Responsável	Usuário responsável pela entrada do requisito	Texto
Status	Situação de execução do requisito	Texto
Prioridade	Grau de prioridade de execução	Texto
DataCriação	Data de criação do requisito	Data
DataModificação	Data de atualização do requisito	Data

Classe: CasosdeTeste

Atributo	**Descrição**	**Tipo**
# Cd_CasodeTeste	Código do caso de Teste	Texto
# Cd_Requisito	Código do requisito associado ao caso de teste	Texto
Nome	Nome Caso de teste	Texto
Descrição	Informa o objetivo do caso de teste	Texto
Tipo	Tipo do caso de teste	Texto
Responsável	Usuário responsável pela entrada do caso de teste	Texto
Riscos	Informa os riscos associados	Texto
Funcionalidade	Define qual a funcionalidade a ser testada	Texto
Prioridade	Grau de prioridade de execução	Texto
DataCriação	Data de criação do caso de teste	Data
DataModificação	Data de atualização do caso de teste	Data

Classe: TesteExecutado

Atributo	**Descrição**	**Tipo**
#Cd_TesteExecutado	Código do teste executado	Texto
DataExecução	Data de execução do teste	Data
Resultado	Resultado do teste	Texto
Resumo	Texto indicando considerações de execução	Texto
Versão	Versão de execução	Texto

Classe: Ferramentas

Atributo	**Descrição**	**Tipo**
# Cd_Ferramenta	Código da ferramenta	Texto
Nome	Nome da ferramenta	Texto
Tipo	Especifica o tipo da ferramenta	Texto
Descrição	Breve texto de descrição	Texto
Versão	Versão da ferramenta	Texto
Anofab	Ano de fabricação	Texto
Integrada	Especifica se ferramenta é integrada	Texto
ExtensãoArquivo	Informações de extensões de arquivos	Texto

Classe: TesteDesempenho

Atributo	**Descrição**	**Tipo**
# Cd_TestDes	Código do Teste de Desempenho	Texto
TempoResposta	Tempo de resposta ao usuário	Numero
SessãoHora	Número de Sessões por hora	Numero
TransaçãoSeg	Transações por segundo	Numero
TaxaTransmissão	Taxa de transmissão	Numero
PáginasSeg	Número de páginas requisitadas por segundo	Numero
TipoTransmissão	Tipo de Transmissão	Texto
UsuariosVirtuais	Numero de usuários virtuais especificados	Numero
DuraçãoTeste	Tempo de duração em teste	Numero
TransaçõesFalha	Numero de transações com erros	Numero
NumErros	Numero de erros na aplicação	Numero
HitsSeg	Numero *hits* por segundo	Numero
Throughput	Numero de operações em na unidade de tempo	Numero

Classe: TesteCompatibilidade

Atributo	**Descrição**	**Tipo**
# Cd_TestComp	Código do Teste de Compatibilidade	Texto
SistemaOperacional	Sistema Operacional de teste	Texto
VersaoSO	Versão do Sistema Operacional	Texto
Navegador	Navegador do teste	Texto
VersãoNavegador	Versão do Navegador	Texto
Plug-in	Tipo de Plug-in utilizado na aplicação	Texto
Resultado	Resultados encontrados	Texto
Problemas	Problemas encontrados	Texto
ResoluçãoVideo	Resolução de Vídeo utilizada	Texto
TaxaTransmissão	Valor da taxa de transmissão utilizada	Texto
TipoTransmissão	Tipo de transmissão utilizada	Texto

Classe: TesteNavegação

Atributo	**Descrição**	**Tipo**
#Cd_TestNav	Código do Teste de Navegação	Texto
StatusErro	Informações do Erro encontrado	Texto
TextoLink	Texto do Link	Texto
URL	Endereço do link	Texto
LinksErros	Total de *Links* com Erros	Numero
TotalLinks	Total de *Links* da aplicação	Numero