MATHEUS MORAIS DOS REIS

Orientador: Elton José da Silva
Co-orientador: Ângelo Magno de Jesus

MISTOOL, UMA FERRAMENTA PARA APLICAÇÃO COLABORATIVA DO MÉTODO DE INSPEÇÃO SEMIÓTICA

Ouro Preto
Novembro de 2010

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

MISTOOL, UMA FERRAMENTA PARA APLICAÇÃO COLABORATIVA DO MÉTODO DE INSPEÇÃO SEMIÓTICA

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

MATHEUS MORAIS DOS REIS

Ouro Preto
Novembro de 2010

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

MISTool, uma Ferramenta para Aplicação Colaborativa do Método de Inspeção Semiótica

MATHEUS MORAIS DOS REIS

Monografia defendida e aprovada pela banca examinadora constituída por:

MSc. Elton José da Silva – Orientador
Universidade Estadual de Campinas

Bel. Ângelo Magno de Jesus – Co-orientador
Universidade Federal de Ouro Preto

Me. Fernando Cortez Sica
Universidade de Campinas

Ouro Preto, Novembro de 2010

Resumo

Há anos uma área da computação denominada Interação Humano-Computador (IHC) estuda os fenômenos de interação entre homens e máquinas. Com o surgimento das interfaces gráficas (GUI - Graphic User Interface) em meados dos anos 1960, a IHC se auto estabeleceu como o setor da computação responsável por avaliar como usuários e sistemas computacionais cooperariam entre si. Na época, isto com certeza não foi uma tarefa fácil, pois os desenvolvedores de software com a mente totalmente inserida no contexto matemático da programação não tinham conhecimento de como produzir sistemas elegantes e de fácil utilização (e talvez nem mesmo ferramentas e recursos suficientes para realizá-lo). Com a evolução dos sistemas operacionais, que com o passar de alguns anos deram suportes a janelas e ofereceram uma melhor inteface gráfica com os usuários, uma abordagem séria e eficaz para avaliar a interação humano-computador se fez necessária.

No ano de 1986, uma abordagem de design centrada no usuário foi proposta por Donald Norman[Norman,1986]. A esta abordagem foi dada o nome de Engenharia Cognitiva, que tem como base teórica a Psicologia Cognitiva. Como foi proposta em um momento histórico em que aplicações mono-usuários eram o foco principal de IHC, a Engenharia Cognitiva deixa de responder a alguns problemas encontrados atualmente, em que aplicações multi-usuários tem tido um crescimento notável. Para tentar responder às perguntas que a abordagem de Norman não consegue, surgiu a Engenharia Semiótica.

Ao contrário da Engenharia Cognitiva, que está centrada no usuário, na Engenharia Semiótica o foco está no designer. Saber como e porquê o designer construiu um certo componente, entender e tentar explicar o que o designer tentou passar ao usuário e avaliar a qualidade de sistemas multi-usuários passam a ser de responsabilidade desta nova e emergente área em IHC.

Fundamentado na teoria da Engenharia Semiótica [3], o Método de Inspeção Semiótica (MIS) é um método relativamente novo de avaliação de sistemas interativos. Apoiado no MIS, foi desenvolvido neste trabalho, um ambiente online colaborativo chamado MISTool, para aplicação da avaliação de sites e sistemas web, integrando um grupo de avaliadores e dando uma nova perspectiva para se avaliar aplicações interativas. A possibilidade de termos uma ferramenta de qualidade e inovadora motivou a conclusão este trabalho, de forma que será de grande benefício tanto para a comunidade acadêmica quanto para profissionais do setor

privado.

*Palavras-chave:*Semiotic Inspection.Semiotic Engineering.HCI.Mistool.Collaborative Enviroments.

Abstract

For years, a computational area is defined Human-Computer Interaction (HCI) and it has been studying the phenomena of interaction between men and machines. In 1960, with the emergence of graphic User Interface (GUI), the HCI has established itself as a computation area, which is responsible for assessing the cooperation between users and computer systems. At that time, the development of HCI was not an easy task, because the software developers had their minds inserted into the context of mathematical programming and did not have knowledge of how to produce elegant systems that were easy to use (and perhaps had not resources to accomplish it). With the growth of operating systems, which for years later was supported windows and offering a better graphical interface with the users, a serious approach to evaluate the human-computer interaction was necessary.

In 1986, an approach to user-centered design was proposed by Donald Norman (Norman, 1986). This approach is called Cognitive Engineering, which has the theoretical basis for cognitive psychology. As it was proposed in a historical moment in which single-user applications were the main focus of HCI, Cognitive Engineering stops responding to some problems currently found in multi-user applications that has had remarkable growth. Trying to answer questions that the Norman´s approach can not do, the Semiotic Engineering was emerged.

Unlike the Cognitive Engineering, which focuses on the user, the Semiotic Engineering focus into the designer. Knowing how and why the designer has built a certain component, understanding and trying to explain what the designer tried to pass to the user and evaluate the quality of multi-user systems become the responsibility of this new and emerging area in HCI.

Based on the Theory of Semiotic Engineering [3], the Semiotic Inspection Method (MIS) is a relatively new method to evaluate interactive systems. Supported by MIS, was developed in this paper an online collaborative environment called MISTool for implementation of the evaluation of web sites and systems that integrates a group of evaluators, and gives access to a new perspective to evaluate interactive applications.

*Keywords:*Semiotic Inspection.Semiotic Engineering.HCI.MISTool.Collaborative Enviroments.

Dedico este trabalho ao meu orientador Elton José da Silva e ao meu co-orientador Ângelo Magno de Jesus.

Agradecimentos

Agradeço primeiramente a Deus. Sem Ele e sem Seu apoio seria impossível concluí-lo. Agradeço também à meus pais, pelo apoio incondicional que me deram nesta fase final dos meus estudos. Ao meu orientador Elton José da Silva pelo conhecimento transmitido e que me fez transformar este projeto em realidade. Ao meu co-orientador Ângelo Magno de Jesus por toda a assistência prestada durante o desenvolvimento deste trabalho. Ao professor Dr. David Menotti pelo seu excelente trabalho lecionando a disciplina de Monografia II. E, por fim, à minha namorada Fabíola pelo amor e suporte.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Durante anos, descobrir como melhorar a interação entre homens e máquinas foi foco de estudos de muitos pesquisadores. A relação humano-computador tem sido um problema desde os primórdios da computação. No passado, programadores desenvolviam sistemas complexos e de difícil entendimento. Isto acabou gerando uma frustração em usuários que, ao invés de explorar ao máximo o potencial computacional de um sistema, tinham receios sobre eles e sobre como estes se comportariam. Com o passar dos anos os computadores se tornaram extremamente populares, as linguagens de programação se desenvolveram e as interfaces gráficas evoluiram de comandos complexos para janelas intuitivas. O foco, que antes era em "apenas"desenvolver um sistema tolerante a falhas, passou a ser em como criar bons produtos finais para o usuário. Logo, avaliar a qualidade da interação homem-software sob a perspectiva do usuário e não mais do sistema tem sido um dos principais focos de estudos dos dias atuais.

1.1 Engenharia Cognitiva X Engenharia Semiótica

Sabe-se que a Engenharia Cognitiva [1] estuda a interação direta usuário-sistema, fazendo uma análise de como o usuário está interagindo com o mesmo e retornando então uma resposta qualificando-o positivamente ou não. As abordagens conigtivas tem seu foco no usuário e em como ele compreende um sistema. A Engenharia Cognitiva espera que o usuário crie um modelo mental do sistema baseado em suas experiências anteriores. Por exemplo: a ação de selecionar todo um texto pode ser feita, geralmente, utilizando-se as teclas Ctrl + A. No editor de textos da Microsoft, o Word, esta ação é feita através das teclas Ctrl + T. Isto pode gerar um problema de interação, pois usuários que não têm conhecimento sobre o produto da Microsoft, criam um modelo mental errado do sistema.

Já a Engenharia Semiótica [3], que complementa a Engenharia Cognitiva, tem o foco no designer do sistema, fazendo uma análise sobre qual a melhor maneira de transmitir uma mensagem ao usuário que seja a mais clara possível, visando assim diminuir o esforço cognitivo que o usuário precisará fazer para alcançar seu objetivo.

A Engenharia Semiótica está baseada na Semiótica, disciplina que estuda os signos e seus significados, os sistemas semióticos e de comunicação, bem como os processos envolvidos na produção e na interpretação de signos. Sabendo disso, pesquisadores da área de IHC que trabalham com Engenharia Semiótica desenvolveram um método para a sua aplicação, o Método da Inspeção Semiótica (MIS)[5].

O MIS não é um método de observação direta da interação do usuário com o sistema e, sim, uma análise da interface/interação do ponto de vista da emissão da mensagem de metacomunicação do designer. Para se aplicar o MIS, um avaliador é necessário. O objetivo do avaliador é analisar uma diversidade de "signos"de um sistema, pelo qual um usuário terá acesso, para determinar o quão claro este signo é e, se este foi bem elaborado pelo designer do sistema em questão. Em [2], a análise de um sistema fundamentado pelo MIS consiste em cinco passos centrais :

1. Reconstrução da meta-mensagem designer-usuário baseada na análise em profundidade dos signos metalingüísticos que descrevem ou explicam o sistema, contidos no conteúdo da ajuda online e outros documentos disponíveis (e. g. website, filmes tutoriais, Termos de Serviço, Política de Privacidade etc.).

2. Reconstrução da meta-mensagem designer-usuário baseada na análise em profundidade dos signos estáticos, compostos por elementos das telas de interface e diálogos.

3. Reconstrução da meta-mensagem designer-usuário baseada na análise em profundidade dos signos dinâmicos, compostos por elementos de interação e comportamento do sistema.

4. Uma comparação contrastiva das meta-mensagens definidas nas etapas 1, 2 e 3, analisando se há possibilidade dos usuários atribuírem significados contraditórios ou ambíguos aos signos que constituem essas 3 mensagens.

5. Apreciação conclusiva da qualidade geral da metacomunicação designer-usuário.

A aplicação do MIS pode parecer uma tarefa fácil. Porém, sistemas complexos e de grande porte podem tornar uma avaliação bem mais complicada do que parece.

1.2 Motivação

Por a Engenharia Semiótica ser uma área relativamente nova de estudos, a falta de material de aprendizado, a dificuldade em se ensinar e aprender o MIS e, principalmente, a falta de ferramentas de apoio a este aprendizado estimularam o desenvolvimento deste trabalho.

1.3 Apresentação do Problema

Em IHC, para se avaliar um sistema interativo sob a perspectiva da Engenharia Semiótica, é necessário uma troca de informações muito grande entre os avaliadores. Estas informações muitas vezes têm que ser recolhidas de vários lugares e pessoas diferentes. Trocar emails com anexos, discutir problemas de signos por vários meios de comunicação e criar uma avaliação final, tem sido uma tarefa árdua para profissionais da área de Interação Humano-Computador.

1.4 Proposta de Solução

Propõe-se neste trabalho unificar em um só ambiente todo o processo de avaliação de sistemas interativos. O MISTool, um ambiente colaborativo online, integra todas as fases do MIS, possibilitando um gerenciamento completo e unificado de uma ou mais avaliações. O MISTool não é uma idéia nova. Um protótipo não funcional do sistema já existe e foi desenvolvido em [2]. A idéia deste trabalho foi dar continuidade ao projeto e desenvolver por completo o MISTool como um ambiente funcional e amigável ao usuário. O MISTool é uma ferramenta colaborativa desenvolvida sobre um framework de apoio a comunidades online chamado OriOnGroups.

Em [4], define-se que o OriOnGroups é uma "community network", ou traduzindo para um termo mais amigável em português, é uma ferramenta de apoio a comunidades online. Foi desenvolvido inicialmente no Departamento de Informática da PUC-RIO no ano de 2002. No início, era uma ferramenta simples e com poucas funcionalidades, porém, com o passar dos anos o OriOnGroups foi sofrendo modificações e melhoramentos expressivos foram adicionados. Como pode ser visto em [6], é possível para cada usuário do OriOnGroups criar até 4 perfis distintos de acordo com sua necessidade : Perfil Pessoal, Perfil Acadêmico, Perfil Profissional e Perfil Lazer. Os usuários podem também criar grupos, requisitar entrada em grupos criados por outros usuários, gerenciar seus grupos e muito mais.

Como [4] define, o OriOnGroups é um sistema em camadas (n-Tier). Ou seja, uma camada inferior provê serviços de maior abstração para uma camada superior. Este é um dos principais motivos que tornou o OriOnGroups o sistema escolhido para servir de base para o MISTool. A figura 1.1 mostra como funciona o esquema de camadas n-Tier.

O intuito da utilização do OriOnGroups como base para o MISTool é facilitar e agilizar o processo de desenvolvimento do sistema, uma vez que preocupações como o tratamento de usuários (login, logout e segurança de informações privadas), criação de grupos de discussão, dentre outras funcionalidades, já se encontram implementadas. O OriOnGroups foi totalmente preparado para ser estendido, logo, é correto afirmar que o MISTool será um sistema estendido do OriOnGroups.

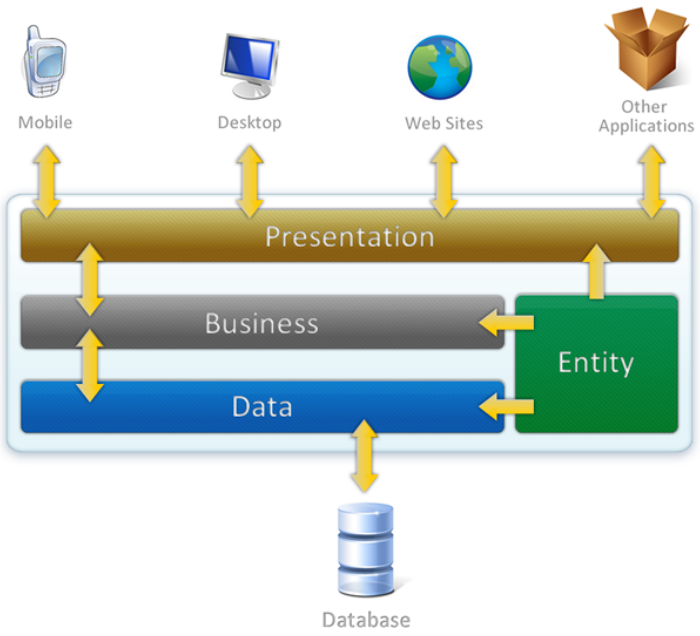

Figura 1.1: Funcionamento de um sistema baseado em camadas n-Tier

1.5 Visão Geral do trabalho

No Capítulo 2 aborda-se o desenvolvimento do MISTool, relacionando sua estrutura com a do OriOnGroups e exemplificando todo o processo de construção do sistema.

Um exemplo da utilização do MISTool é dado no Capítulo 3, que exibe a criação de uma inspeção e a avaliação de signos fictícios.

O Capítulo 4 trata do resultado obtido neste projeto (MISTool), aborda a possibilidade de trabalhos futuros e finaliza este trabalho apresentando uma conclusão.

Capítulo 2

Desenvolvimento

O MISTool foi desenvolvido como uma extensão do OriOnGroups. Logo, é de se esperar que para facilitar a integração entre estes dois sistemas, o produto deste trabalho siga a mesma sistemática estrutural que o seu sistema base. O desenvolvimento do MISTool pode ser resumido em seis fases principais:

1. Preparação do ambiente e análise inicial
2. Arquitetura do projeto
3. Modelagem dos casos de uso
4. Modelagem de dados e Implementação
5. Criação do documento de ajuda
6. Teste do sistema

Na fase 4, durante a fase de implementação, várias etapas foram feitas várias vezes. São elas:

- ETAPA 1: Modelagem de tabelas do banco de dados
- ETAPA 2: Integração com o OriOnGroups
- ETAPA 3: Modelagem das classes necessárias ao MISTool
- ETAPA 4: Modelagem e criação da camada de negócios
- ETAPA 5: Criação da camada de apresentação
- ETAPA 6: Testes locais

As etapas da fase de implementação são independentes entre si. Por exemplo: mesmo estando o desenvolvedor a trabalhar na criação da camada de apresentação, este pode ter que voltar em uma etapa anterior para aplicar alguma modificação necessária no sistema. Isto não significa também, que o desenvolvedor tenha que refazer todas as etapas novamente para voltar a trabalhar de onde parou. A Figura 2.1 ilustra o processo básico de desenvolvimento da implementação do MISTool.

Figura 2.1: Ilustração da etapa de implementação. Cada aresta representa um caminho de duas vias.

A seguir, será dada uma explicação detalhada de todo o processo de desenvolvimento do MISTool.

2.1 Preparação do ambiente e análise inicial

Inicialmente, definiu-se a linguagem de programação adotada. Utilizou-se PHP 5 [7] para a construção deste projeto. Porém, o OriOnGroups foi desenvolvido em PHP 4, o que poderia causar algum tipo de instabilidade no código, mas isto não aconteceu devido ao suporte que PHP dá à suas versões antigas. Com a linguagem definida, o próximo passo foi definir o ambiente de programação. Devido a experiências anteriores o Eclipse Galileo [10] com o plugin PHPEclipse foi o escolhido.

O servidor web e o banco de dados foram escolhas fáceis de serem feitas. Apache 2.2 [8] foi escolhido como o servidor web por ser grátis e bastante difundido entre os programadores PHP. O banco de dados utilizado foi MySQL [9], também por ser grátis e ter uma grande comunidade de usuários ativa. Mas, o principal motivo que determinou a escolha do MySQL foi que o OriOnGroups também o utiliza. Ou seja, o fator determinante na escolha dessas ferramentas foi a facilidade com que elas provém a integração OriOnGroups-MISTool.

O framework escolhido para tratar da camada de apresentação foi o Smarty 2.6 [11]. O OriOnGroups utiliza o Smarty 1.5, que se tornou obsoleto com o passar dos anos. Como

podem existir várias instâncias do Smarty para um mesmo sistema, não houve conflitos entre as duas versões, apenas algumas poucas adaptações no código do Orion foram necessárias.

2.2 Arquitetura do projeto

A estrutura do MISTool é bastante parecida com a do OriOnGroups. Para facilitar a manutenção de ambos os sistemas, a divisão em camadas n-Tier foi adotada, a nomenclatura de pastas e arquivos e os padrões de projeto também. No restante deste capítulo, toda a estrutura do MISTool será apresentada em detalhes.

2.2.1 Tabelas

As tabelas do MISTool se localizam dentro do banco de dados do OriOnGroups. Criar um novo banco para o projeto foi uma idéia descartada de início, pois isto apenas aumentaria o grau de dificuldade do desenvolvimento, dado que o Orion já contém todos os métodos de acesso ao banco. Para diferenciar as tabelas dos dois sistemas, uma nomenclatura simples foi adotada, sendo que as tabelas do MISTool têm um prefixo "mis_"no início. Mais adiante, será mostrado como estão modeladas as tabelas do MISTool.

2.2.2 Banco

Para se conectar ao banco de dados, o MISTool reutiliza o pacote Banco do OriOnGroups. Este pacote é responsável por estabelecer a conexão com o banco de dados.

2.2.3 DAO

As classes contidas no pacote DAO (Data Access Object) são responsáveis por realizar consultas, determinar como os dados sao inseridos, alterados ou deletados e retornar estas informações ao sistema. As informações recuperadas, são então transformadas em objetos, isto agiliza o desenvolvimento pois objetos são facilmente manipulados pelo PHP, dimuindo assim o número de consultas necessárias para se obter um dado do banco. A Figura 2.2 mostra um exemplo de código dos arquivos DAO.

```
public static function inserir_usuario_inspecao($cod_insp, $cod_usuario){

    $banco = Banco::getBanco();

    $sql = "
            insert into mis_avaliadores
                (cod_usuario,cod_grupo)
            values
                (
                    '$cod_usuario',
                    '$cod_insp'
                    )
    ";
    $sucesso = $banco->executar($sql);
    if($sucesso == FALSE){
        return FALSE;
    }
    return "SUCESSO";
}
```

Figura 2.2: Exemplo de código DAO

Como pode ser visto na Figura 2.2, o método da classe DAO em si não faz diretamente uma conexão com o banco de dados. Utilizou-se uma chamada ao método getBanco() da classe Banco para realizar a conexão. Esta abstração é importante para manter a robustez do sistema.

2.2.4 Base

No pacote BASE, assim como no OriOnGroups, estão as classes que determinam o funcionamento do nosso sistema, tais como Inspeção e Signo. Essas classes não detêm nenhum conhecimento sobre as consultas ao banco. Para fazer as consultas necessárias, as classes deste pacote são apoiadas pelas classes DAO. Ou seja, quando elas precisam realizar qualquer atividade no banco de dados, elas requisitam às classes do pacote DAO que façam estas tarefas.

2.2.5 PHP

Neste pacote está toda a lógica do sistema. Os arquivos PHP são responsáveis por realizar e atribuir funções, tais como: Disponibilizar variáveis para a camada de apresentação, verificar segurança e etc.

2.2.6 Templates

Diferentemente do OriOnGroups, as páginas da camada de visualização do MISTool localizam-se no pacote de templates e suas extensões não são .html, e sim .tpl. Estas são as páginas que serão exibidas para o usuário e este pacote constitui a última camada de abstração baseado no esquema n-Tier.

2.2.7 CSS, JS e Images

Talvez o único grande problema do OriOnGroups foi não ter dado a devida atenção aos pacotes CSS, JS e Images. Estes pacotes representam todo o tratamento de layout de nosso sistema. Ou seja, são eles que tornam as páginas do MISTool atrativas e bonitas à vista do usuário. Muito bem elaborados em nosso sistema e utilizando o que há de mais novo em questão de estilo, estes pacotes estão em constante mudança, pois a cada nova linha adicionada nos arquivos de template estes pacotes sofrem alterações.

2.3 Modelagem dos casos de uso

Poucas pessoas e até mesmo poucas empresas de desenvolvimento de software dão alguma importância para a modelagem de casos de uso. Porém, os casos de uso dão uma visão geral e relevante sobre como o usuário irá interagir com o sistema. Saber quais são as possíveis ações a partir do local em que o usuário se encontra pode levar ao programador a prever determinadas situações e até mesmo evitar erros que poderiam facilmente passar despercebidos.

Para o MISTool, foram desenvolvidos três modelos de casos de uso, o primeiro é um modelo básico e geral que aborda as principais ações do usuário no sistema. O segundo é um modelo mais específico relacionado com as inspeções, ou seja, todas as possibilidades existentes de relação entre usuário-inspeção foram aqui mapeadas. O terceiro e último modelo refere-se às ações dentro das inspeções, ou seja, na criação das cinco etapas do MIS.

2.3.1 Modelo de caso de uso geral

O modelo de caso de uso geral é um mapeamento superficial do sistema. Isto quer dizer que as principais ações disponíveis para o usuário ao entrar no sistema estão modeladas aqui. A Figura 2.3 oferece uma informação visual sobre o que foi explicado.

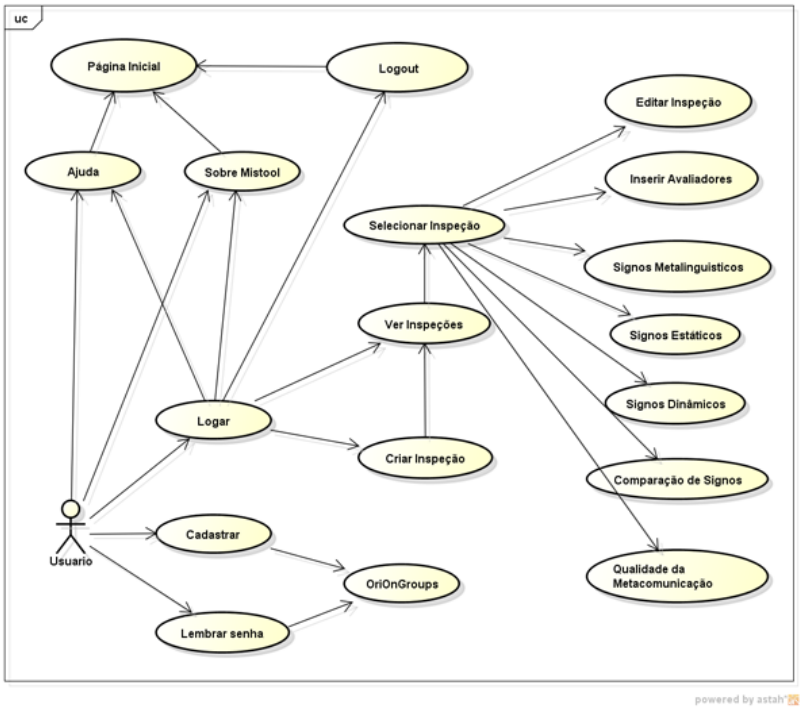

Figura 2.3: Modelo do caso de uso principal do MISTool.

Inicialmente, o usuário tem cinco opções possíveis no sistema : Efetuar o login, Cadastrar, Lembrar senha e login, ver informações sobre o MISTool e ver o documento de ajuda. Se o usuário optar por cadastrar ou lembrar a senha e login, ele será redirecionado para essas respectivas opções no OriOnGroups, uma vez que o MISTool apenas reutiliza o sistema de controle de usuário. Se optar por ver informações sobre o MISTool ou ver o documento de ajuda, o usuário tem a opção de voltar a página inicial. Neste momento, fechamos o primeiro ciclo de interação do sistema.

2.3.2 Modelo de caso de uso de inspeções

Ao optar por fazer o login no sistema, inicia-se o segundo ciclo. Agora, novas opções estão disponíveis. Além de poder acessar o documento de ajuda e as informações sobre o MISTool o usuário pode criar inspeções, ver as inspeções que participa e sair do sistema. Se o usuário optar por sair do sistema ele é redirecionado para a página inicial.

Pela Figura 2.4 (que ilustra os casos de uso relacionados à inspeção), se o usuário escolher criar uma nova inspeção, em caso de sucesso ele é redirecionado para a página 'Ver Inspeções que participo'. Se ocorrer algum tipo de erro ao criar uma nova inspeção, o sistema lhe retorna para a mesma página explicando qual o erro que ocorreu. Caso o usário clique em 'Ver Inspeções que participo', ele poderá excluir as inspeções que é dono, ou selecionar alguma inspeção. Se clicar em excluir, ele será redirecionado novamente para a página que estava.

Caso clique para ver alguma inspeção, novas opções tornam-se disponíveis para ele, tais quais: Editar Inspeção, Inserir avaliadores, Signos Metalinguísticos, Signos Estáticos, Signos Dinâmicos, Comparação de Signos e Qualidade da Metacomunicação. Se clicar em Editar Inspeção, o usuário poderá editá-la ou remover avaliadores que participam nela. Em caso de sucesso na operação desejada, em ambos os casos o sistema redireciona o usuário para a página em que se encontrava. Caso um erro aconteça, o usuário é também redirecionado à pagina em que estava e uma mensagem de erro aparece na tela.

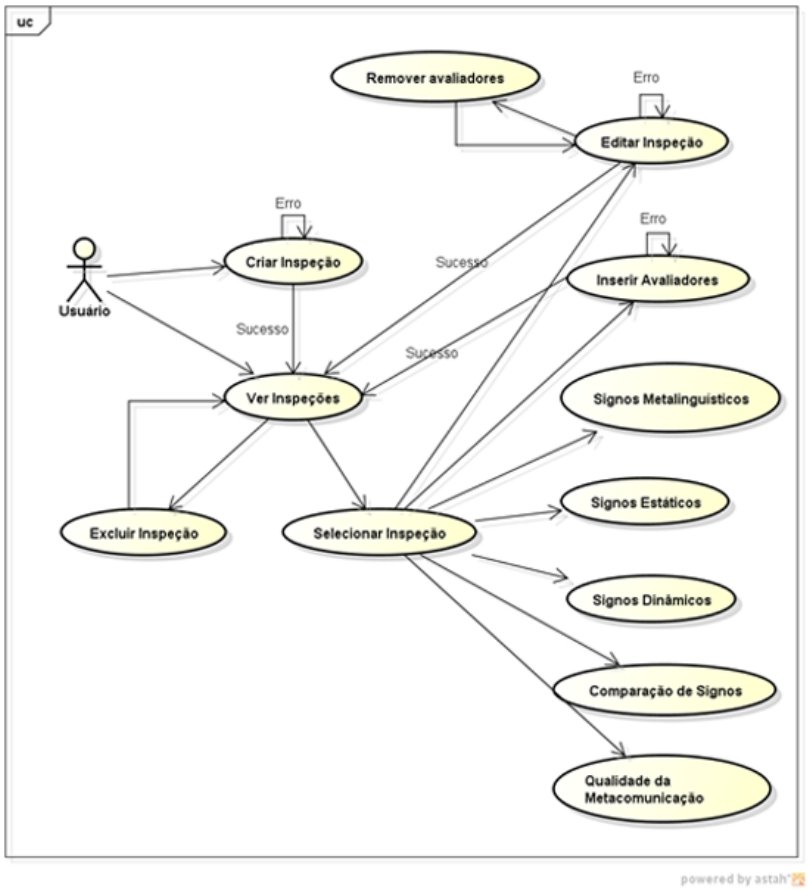

Figura 2.4: Modelo do caso de uso de inspeções.

2.3.3 Modelo de caso de uso de Signos

O terceiro e último de ciclo dos modelos de casos de uso do MISTool, é mostrado na Figura 2.5.

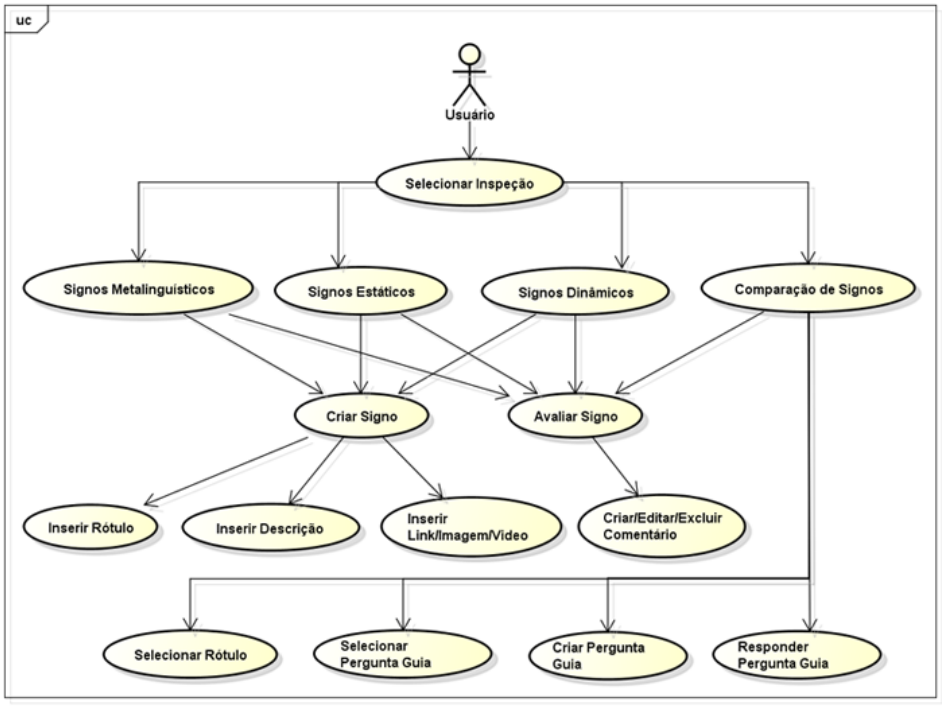

Figura 2.5: Modelo do caso de uso de signos.

Ao selecionar uma inspeção as opções disponíveis relacionadas com signos, são: Signos Metalinguísticos, Signos Estáticos, Signos Dinâmicos, Comparação de Signos e Qualidade da Metacomunicação. Qualidade da Metacomunicação não aparece na Figura 2.5 porque ela não estará disponível nesta primeira versão do sistema do MISTool. Se o usuário clicar em Signos Metalinguísticos, Signos Estáticos ou Signos Dinâmicos, estará disponível para ele criar um novo signo ou avaliar um signo existente. Para criar um novo signo o usuário deverá inserir um rótulo, uma descrição e, se ele estiver em Signos Metalinguísticos, um link. Se o usuário estiver em Signos Estáticos, poderá adicionar uma imagem e se o usuário estiver em signos dinâmicos poderá adicionar um vídeo. Em Avaliar Signo, o usuário poderá inserir um comentário sobre um signo existente, editar um comentário ou excluir um comentário previamente inserido por ele. Em Comparação de Signos, as seguintes opções estarão disponíveis: selecionar o rótulo a ser comparado, criar perguntas guias, selecionar perguntas guias e responder à pergunta guia selecionada.

2.4 Modelagem de dados e Implementação

A modelagem de dados, ao contrário da modelagem de casos de uso, é uma tarefa fundamental no desenvolvimento de qualquer sistema. Uma modelagem ruim dos dados das tabelas no banco de dados e do relacionamento entre estas tabelas, podem levr o sistema a um fraco desempenho. Os dados das tabelas estão armazenados em algum disco rígido e a consulta deste

disco é o processo mais lento para se acessar dados arquivados. Logo, uma modelagem ruim, pode levar o sistema a realizar mais consultas no disco rígido, deixando-o lento e prejudicando as ações do usuário. A fase de implementação também é extremamente importante. Um algoritmo mal elaborado pode exigir muitas consultas ao banco prejudicando o desempenho do sistema e a sua robustez, deixando-o assim, com maior possibilidade de ocorrerem erros durante as ações do usuário. Nas próximas seções, será dada uma visão geral da estrutura das tabelas do OriOnGroups e do MISTool e como elas estão relacionadas e será mostrada como foi abordada a implementação do sistema deste trabalho.

2.4.1 Estrutura de tabelas do OriOnGroups e MISTool

As modelagens de dados do MISTool ocorreram em duas fases. Na fase inicial, criou-se um modelo superficial com o foco nas tabelas do OriOnGroups. Na fase final o foco foi na criação das tabelas do sistema e como realizar as integrações entre OriOnGroups e MISTool.

2.4.1.1 OriOnGroups

Para entender as tabelas do MISTool, precisa-se conhecer as tabelas do seu sistema base. As Figuras 2.6, 2.7 e 2.8, retiradas de [4], exibem uma noção de parte da estrutura do OrionGroups.

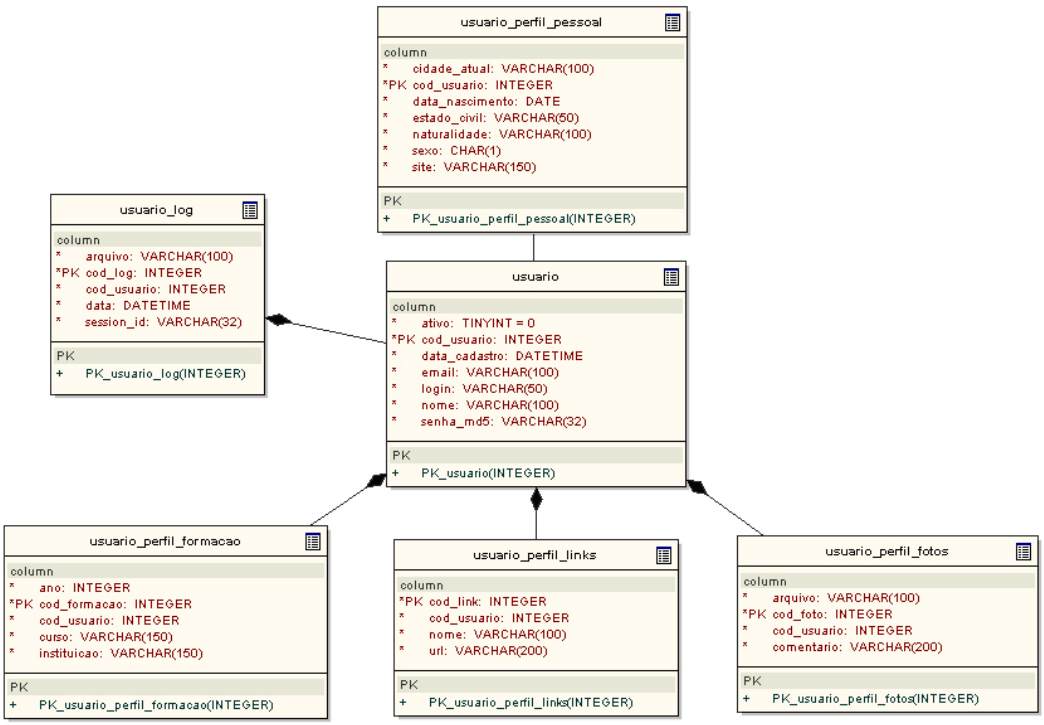

Figura 2.6: Tabela usuario e suas relações.

Na figura 2.6, a única tabela que é do nosso interesse é a tabela usuario, pois as outras tabelas que têm relação com usuario são de uso apenas do OriOnGroups e não alteram em nada o funcionamento do MISTool. É utilizando esta tabela (usuario), recorrendo a chamadas de funções do OriOnGroups ao banco que controlamos como os usuários executam, ou podem executar, ações no MISTool.

A figura 2.7 apresenta a tabela grupo e suas relações. A relação que esta tabela tem com a tabela usuário é uma das relações do OriOnGroups que afeta diretamente o MISTool, isto porque a classe Inspeção no MISTool estende a classe Grupo do Orion. Logo, afirmar que a classe Grupo tem uma relação com a classe Usuario é o mesmo que afirmar que a classe Inspeção tem uma relação (não direta) com a classe Usuario. Isto será explicado melhor na seção 2.4.3.

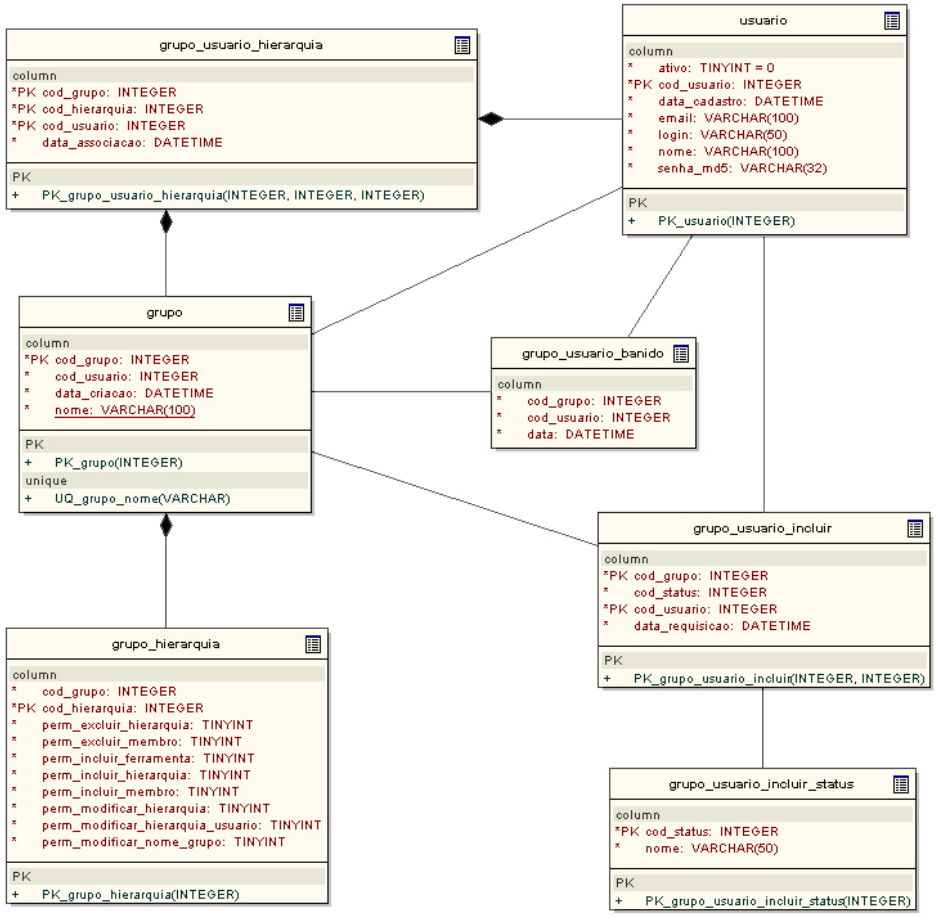

Figura 2.7: Tabela grupo e suas relações.

A última e não menos importante classe do OriOnGroups que tem ligação com o MISTool é a classe Fórum. Como pode ser visto na Figura 2.8, a tabela grupo_forum relaciona-se com as tabelas grupo e usuario. Cada grupo pode ter um fórum, fóruns possuem temas,

temas possuem assuntos e cada assunto pode conter várias árvores de discussões [Silva, 2006]. No caso do MISTool uma Inspeção "é um"Grupo. Logo, ela contém um Fórum. Dentro do sistema definiu-se que um tema será sempre o nome da inspeção criada. O assunto será sempre o rótulo da inserção de um novo signo e as falas representarão as discussões entre os avaliadores.

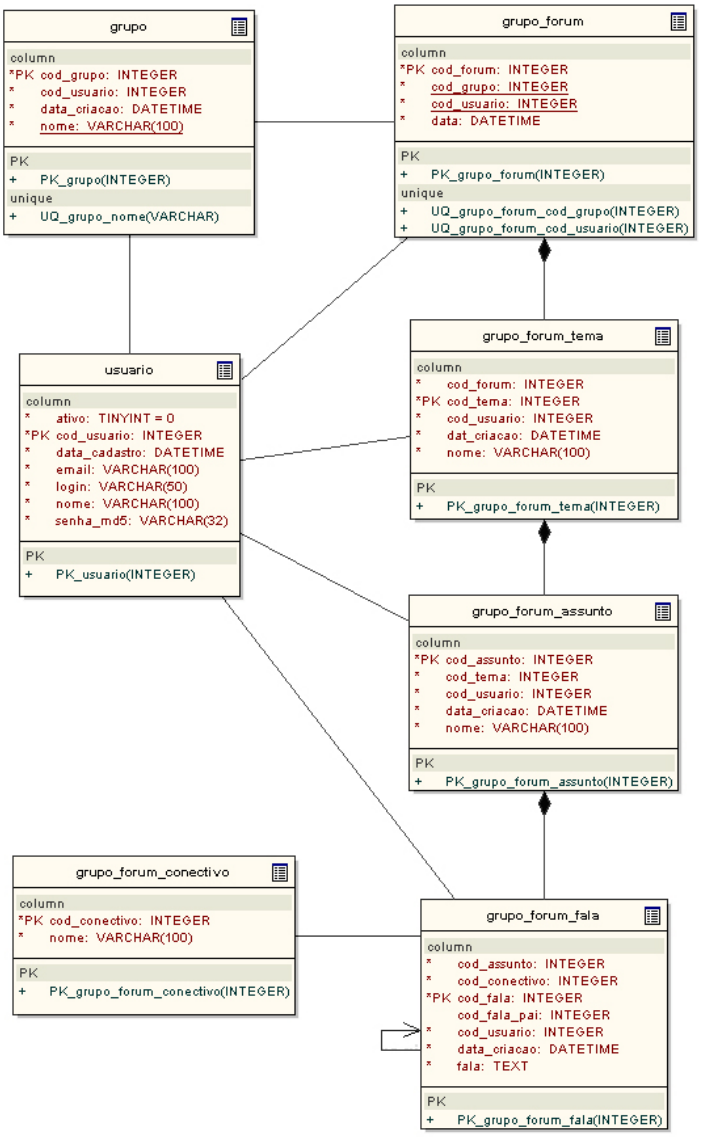

Figura 2.8: Tabela forum e suas relações.

2.4.1.2 MISTool

Após uma análise da estrutura do OriOnGroups e de suas tabelas úteis para o MISTool, modelou-se as tabelas do nosso sistema conforme a figura 2.9 apresenta.

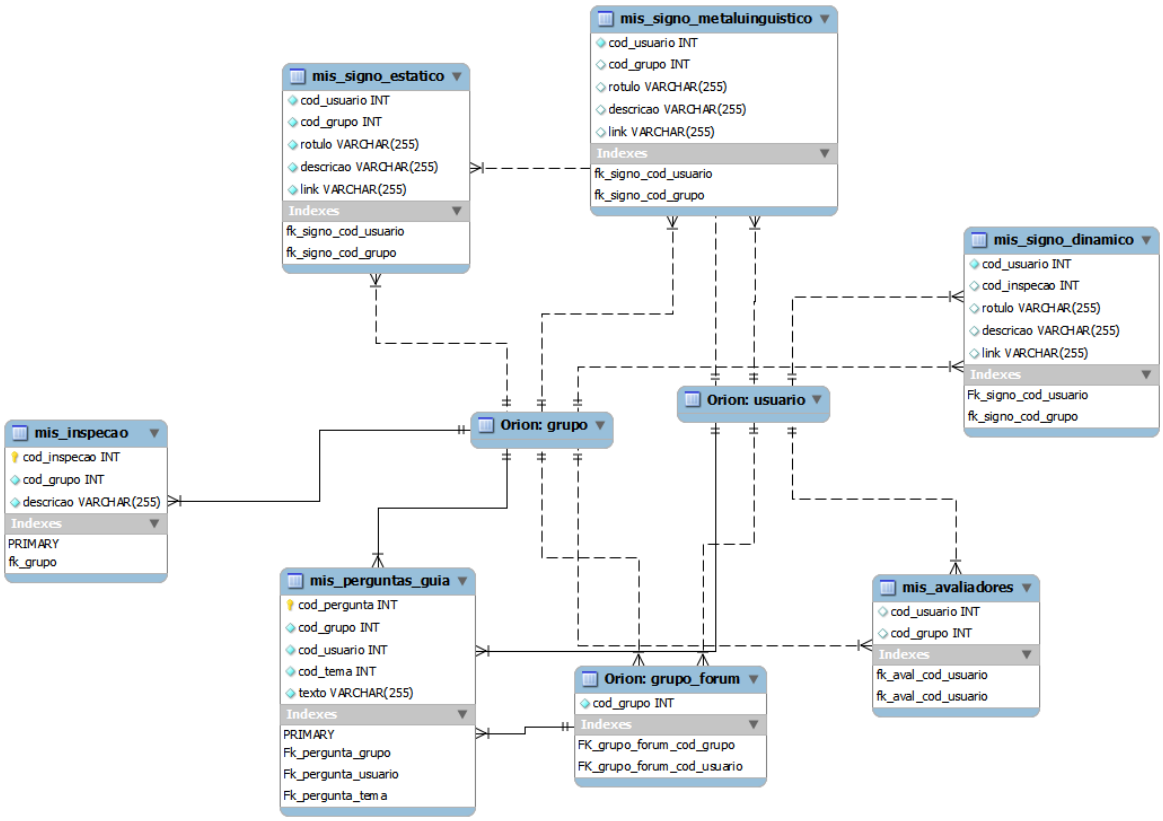

Figura 2.9: Tabela Inspeção e suas relações.

Na figura 2.9, as tabelas que iniciam o nome com "Orion:"são tabelas do OriOnGroups e as tabelas que iniciam com "mis_"são tabelas do MISTool. Nota-se a relação entre as tabelas mis_inspecao e grupo. Como visto anteriormente, Inspeção "é um"Grupo e a relação entre as classes se reflete no banco.

A tabela mis_avaliadores é responsável por armazenar o código do usuário e o código do grupo. Com esses dados pode-se saber quem participa em qual inspeção como avaliador.

Ao inserir um signo metalinguístico, estático ou dinâmico, os dados destes signos ficam armazenados, respectivamente, nas tabelas: mis_signo_metalinguistico, mis_signo_estatico e mis_signo_dinamico. O campo "link"nestas tabelas armazena o endereço de links (signos metalinguísticos), imagens (signos estáticos) e vídeos (signos dinâmicos). Essas tabelas têm relação com grupo e usuario. Esta relação serve para identificar quem criou o signo e em qual inspeção ele foi criado.

Por fim, a tabela mis_perguntas_guia armazena as perguntas guias que serão utilizadas quando a avaliação estiver na etapa de comparação de signos. Esta tabela tem relação com grupo, usuario e grupo_forum. Estas informações são necessárias para que o sistema saiba a qual inspeção, usuário e fórum uma determinada pergunta guia pertence.

2.4.2 Implementação do MISTool

O desenvolvimento do MISTool ocorreu de forma rápida e prática. Rápida porque o OriOn-Groups já continha quase todas as ferramentas necessárias para o MISTool, e prática porque os frameworks utilizados em sua construção agilizaram o processo de produção. O resultado disto é um código limpo, simples, eficiente e de fácil manutenabilidade. Com a separação da lógica do sistema e da apresentação do mesmo, um designer pode facilmente reconstruir todo o layout sem a necessidade de executar mudanças em nenhum arquivo que trabalha com o funcionamento do MISTool.

A preocupação inicial no desenvolvimento do MISTool foi com o layout. Produzir o layout e um esquema de cores adequados são procedimentos chave para que o resultado final do projeto seja satisfatório principalmente para o usuário do sistema.

2.4.2.1 Construção do Layout

Um modelo de layout é desenvolvido visando sempre uma maneira de facilitar ao usuário identificar os possíveis ações que ele pode tomar no sistema. O modelo de layout que se adequou muito bem no MISTool é mostrado na Figura 2.10.

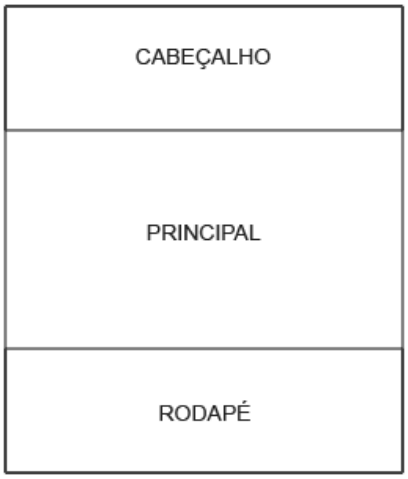

Figura 2.10: Layout escolhido para o MISTool.

Este layout indica que nosso site está divido em três principais camadas, que são: Cabeçalho, Principal e Rodapé. Cabeçalho e Rodapé são estáticos no sistema. A camada Principal é dinâmica e é a camada que oferece interatividade ao usuário. Na implementação do mistool, foi criada uma camada a mais que engloba as três camadas citadas. Esta camada é responsável por centralizar o conteúdo do sistema no navegador do usuário independente de sua resolução de vídeo e chama-se WRAP. A estrutura principal do MISTool fica da seguinte forma:

- <INICIO_WRAP>

  <Cabeçalho>

  <Principal>

  <Rodapé>

- <FIM_WRAP>

Após definir o modelo de layout, projetou-se o cabeçalho e o rodapé. Ambos são inseridos sempre que uma nova página é criada, o cabeçalho é o primeiro a ser adicionado e o rodapé o último.

A última observação a ser feita sobre o layout do MISTool é que todas as suas camadas flutuam para a esquerda. Sistemas bem definidos que são construídos com base em línguas ocidentais, geralmente utilizam esta técnica. Isto facilita na estilização de componentes e camadas no CSS e na organização estrutural do sistema.

A fase final da construção do layout inclui o desenvolvimento do Menu. Os menus do MISTool sâo dinâmicos, ou seja, dependendo da ação do usuário, novas opções são oferecidas. Desta forma, por questões de segurança, a inserção do menu é feita dentro da camada Principal do sistema.

Após ter definido toda a estrutura de layout do MISTool, inicia-se a fase de desenvolvimento de páginas interativas.

2.4.3 Desenvolvimento de páginas interativas

Nesta seção, será abordada de forma prática como foi feito o desenvolvimento das páginas interativas do MISTool. Um exemplo de todo o ciclo de criação de uma Inspeção será mostrado para exemplificar toda a construção deste projeto.

Inicialmente, definiu-se que a classe Inspeção do MISTool é uma extensão da classe Grupos do OriOnGroups, logo, iniciou-se por ela. Criou-se a classe Inspeção dentro do pacote BASE do nosso sistema pois é ele que é o responsável por armazenar todas as classes do MISTool. A Figura 2.11, mostra a classe Inspeção, seus atributos e métodos.

```php
<?php
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/php/base/Grupo.php");

class Inspecao extends Grupo{

    private $descricao;

    public function __construct()
    {
        parent::__construct();
    }


    public function setDescricao($newVal)
    {
        $this->descricao = $newVal;
    }

    public function getDescricao()
    {
        return $this->descricao;
    }
}
?>
```

Figura 2.11: Ilustração da classe Inspeção.

Por estender a classe Grupo, na linha 1 requisita-se ao PHP que inclua esta classe junto à classe Inspeção. Assim, torna-se possível acessar os métodos de Grupo a partir de um objeto Inspeção.

Finalizada a criação da classe, o objetivo agora é criar a tabela qua armazenará as inspeções criadas e suas relações. Como a classe Inspeção possui apenas um atributo, não mais do que três campos na tabela serão necessários para guardar as informações que o sistema necessita. O nome dado para a tabela que será criada no banco é mis_inspecao. Nesta tabela cria-se os campos: cod_inspecao, cod_grupo, e descricao. O campo cod_inspecao é um identificador único que cada inspeção receberá ao ser criada. Diferentemente, o campo cod_grupo foi definido como sendo uma chave estrangeira da tabela e se relaciona com o campo cod_grupo da tabela grupo. Isto se faz necessário, pois quando cria-se uma inspeção cria-se também um grupo, e deve-se ter um valor único que possa relacioná-los. A relação com a tabela usuario não foi feita pois se for necessário descobrir quem criou a inspeção, simplesmente acessa-se a tabela grupo e requisita o código do usuário. A Figura 2.12 mostra o resultado final da tabela mis_inspecao.

	Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ação
☐	cod_inspecao	int(11)			Não		auto_increment	
☐	cod_grupo	int(11)			Não			
☐	descricao	varchar(255)	latin1_general_ci		Não			

Figura 2.12: Tabela mis_inspecao.

A esta altura, tem-se a classe Inspeção e a tabela com todos os seus relacionamentos criados. Porém, o usuário ainda não tem como criar inspeções pois não foram criados os meios que o possibilitam executar esta ação. A maneira mais simples encontrada foi criar um menu

que ao logar-se no sistema disponibiliza os links para o usuário. Construiu-se então a parte visual deste menu. Note que até o momento apenas a camada da lógica de negócios de nosso sistema foi abordada. Agora, o trabalho será realizado também na camada de apresentação. Criou-se o arquivo menu_principal.tpl dentro do pacote de templates do MISTool. A extensão tpl é padrão para templates gerenciados pelo Smarty e aceita tanto códigos html como php. O Smarty trabalha também com atribuições de variáveis e disponibilização destas para o template. A Figura 2.13, mostra o código do menu criado.

```
<div class="separator">
    <div class="sub_item_menu sair">
        <div class="arrow ball"></div>
        <a href="../index.php">Sair do Mistool</a>
    </div>
    <div class="sub_item_menu">
        <div class="arrow"></div>
        <a href="../php/ajuda.php">Ajuda</a>
    </div>
    <div class="sub_item_menu">
        <div class="arrow"></div>
        <a href="../php/sobre_mistool.php">Sobre o Mistool</a>
    </div>
    <div class="sub_item_menu">
        <div class="arrow"></div>
        <a href="../php/inspecao_participa.php">Ver inspeções que participo</a>
    </div>
    <div class="sub_item_menu">
        <div class="arrow"></div>
        <a href="../php/inspecao_criar.php">Criar nova inspeção</a>
    </div>
</div>
```

Figura 2.13: Código do menu principal do sistema.

Apenas o código html é suficiente mas não satisfatório. Então, pode-se observar a criação de divs no menu. As divs e a definição de classe para as mesmas servem para que o arquivo CSS consiga identificar qual estilo aplicar em qual componente. Criou-se então os estilos desejados que se aplicam ao menu. Um exemplo do código css aplicado ao menu principal aparece na figura 2.14.

```
71 div.wrap div.separator {
72     background-color:#FFF;
73     width:900px;
74     height:25px;
75     float:left;
76     border-top:1px dashed #2c539e;
77     border-bottom:1px dashed #2c539e;
78 }
79
80 div.wrap div.separator div.sub_item_menu {
81     float:right;
82     height:24px;
83 }
84
85 div.img,div.wrap div.separator div.sub_item_menu.sair{
86     float: left;
87 }
88
89 div.wrap div.separator div.sub_item_menu a {
90     color:#2c539e;
91 }
92
93 div.wrap div.separator div.sub_item_menu div.arrow {
94     background-image:url("../images/menu/arrow.jpg");
95     background-repeat:no-repeat;
96     float:left;
97     width:14px;
98     height:20px;
99     margin-left:10px;
100    margin-top:4px;
101 }
```

Figura 2.14: Exemplo do código CSS aplicado ao menu principal.

Como a classe "separator"é única no sistema, uma chamada do tipo div.wrap div.separator é suficiente para se alterar o seu layout. Se houvessem mais classes com este nome, um caminho mais específico se faria necessário, como por exemplo: div.wrap div.main div.classeDaDiv div.separator. A Figura 2.15 ilustra o menu principal finalizado.

Figura 2.15: Ilustração do menu principal.

No espaço individual do usuário (que é executado após o login no sistema), solicita-se ao Smarty que inclua o menu que foi criado. Agora que o usuário tem os meios físicos necessários para criar uma inspeção, volta-se a camada de lógica de negócios e define-se para o sistema como isto ocorre. Pela Figura 2.13, na linha vinte indicou-se que se o usuário clicar em "Criar nova inspeção", então, este será direcionado para o arquivo inspecao_criar.php. Logo, o próximo passo é desenvolver este arquivo. Por fazer parte da lógica do sistema, ele arquivo foi criado dentro do pacote PHP. A Figura 2.16 exemplifica o seu código.

```php
<?php
session_start();

require('../smarty_connect.php');
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/php/base/Usuario.php");

$smarty = new smarty_connect;

$usuario = Usuario::getLogado();

if($usuario == FALSE){
    header("Location: http://".$_SERVER['HTTP_HOST']."/oriongroups/mistool/index.php");
}

$smarty->assign('usuario',$usuario);

$smarty->display('inspecao_criar.tpl');
?>
```

Figura 2.16: Código do arquivo inspecao_criar.php.

Nas linhas 4 e 5 da figura acima, é feita uma requisição ao php para requerer os arquivos smarty_connect.php e Usuario.php. O primeiro tem as funções necessárias para se criar uma instância do Smarty. O segundo é a classe Usuario do OriOnGroups, com ela pode-se verificar, por exemplo, se o usuário está logado ou não no sistema. Na linha 7, cria-se um objeto do Smarty. Na linha 9 atribui-se à variável usuario o resultado da chamada do método getLogado() da classe Usuário. Nas linhas 11, 12 e 13, verifica-se se o usuário está logado. Se não estiver, redireciona-o para a página inicial do MISTool. Na linha 15, a chamada ao método "assign"do Smarty é feita e uma atribuição é executada. Atribuiu-se para uma nova variável chamada também de "usuario"o valor do objeto usuario que foi armazenado anteriormente. O que isto quer dizer é que esta nova variável usuário estará disponível para ser usada no template que o Smarty designar. Finalmente, na linha 17 indica-se ao sistema que a página que será exibida como resultado é um template do Smarty e que se chama inspecao_criar.tpl.

O passo seguinte é criar o template (a página) que será exibida ao usuário após clicar em "Criar Inspeção"no menu principal. A Figura 2.17 exemplifica o código do template inspecao_criar.tpl.

```
1{include file="header.tpl" title="Mistool" css="/oriongroups/mistool/css/mistool.css"}
2{include file="menu_principal.tpl"}
3<div class="titulo">
4    <img src="../images/titulos/novaInspecao.png" width="500" height="40"></img>
5</div>
6
7<div class="formulario">
8    <form class="formulario" name="formNovaInspecao" method="post" action="../php/inspecao_criar_enviar.php">
9        <div class="entrada">
10           <div class="campo_entrada"> Título: </div> <input id="form_titulo" class="form" type="text" name="titulo">
11           <div class="erroLogin">{if ($titulo_branco == 'erro')}
12               <div class="erro">Por favor, digite um título</div>
13           </div>
14           <div align="center">{/if}</div>
15           <div class="erroLogin">{if ($nome_erro == 'erro')}
16               <div class="erro">O nome digitado é inválido</div>
17           </div>
18           <div align="center">{/if}</div>
19       </div>
20       <div class="entrada">
21           <div class="campo_entrada"> Descrição: </div> <textarea name="desc" rows="5" cols="26" wrap="auto"></textarea>
22           <div class="erroLogin">{if ($desc_branco == 'erro')}
23               <div class="erro">Por favor, digite uma descrição.</div>
24           </div>
25           <div align="center">{/if}</div>
26       </div>
27       <div class="btn_submit">
28           <input type="submit" name="Submit" value="Criar">
29       </div>
30   </form>
31</div>
32{include file="footer.tpl"}
```

Figura 2.17: Código do template inspecao_criar.tpl.

A primeira linha da Figura 2.17 solicita ao Smarty a inclusão do arquivo header.tpl, que é o cabeçalho do MISTool. Da mesma forma, a segunda linha solicita o arquivo menu_principal.tpl. Entre as linhas 3 e 31, definiu-se a estrutura da página, que tem uma figura no inicio e um formulário, que é onde o usuário entrará com os dados da inspeção que deseja criar. Porém, o mais importante a ser notado neste código é o uso de variáveis e a inserção de uma certa lógica no arquivo de template. O Smarty possibilita o uso de lógica no template para que este seja um arquivo mais robusto e fácil de ser administrado. Como pode ser observado na linha 11, o uso de estruturas de controle e variáveis é que torna o Smarty um dos melhores "template engine"que existe para php atualmente. Nesta linha, uma verificação é feita para saber se o usuário digitou um título para a inspeção. De forma semelhante é verificado se o título é único (linha 15) pois não podem existir duas ou mais inspeções com o mesmo título no sistema, e na linha 22 verifica-se se o usuário digitou uma descrição. Uma explicação mais consistente sobre estas variáveis será feita em seguida. Por fim, pode-se observar que o formulário para a criação da inspeção envia os dados inseridos pelo usuário para o arquivo inspecao_criar_enviar.php. O próximo passo é criar esta página.

Novamente, por uma página da camada de negócios, criou-se a página inspecao_criar_enviar.php dentro do pacote PHP do sistema. Como o código desta página é maior e mais complexo, sua explicação será feita em quatro passos. A Figura 2.18 mostra a parte inicial do arquivo.

```php
<?php
session_start();

require('../smarty_connect.php');
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/php/form/FormSecurity.php");
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/mistool/php/base/Inspecao.php");
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/mistool/php/dao/DAOInspecao.php");
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/php/base/Usuario.php");
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/php/dao/DAOPerfilGrupo.php");
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/php/dao/DAOUsuarioPermissaoPerfil.php");

$usuario = Usuario::getLogado();

if($usuario == FALSE){
    header("Location: http://".$_SERVER['HTTP_HOST']."/oriongroups/mistool/index.php");
}

$smarty = new smarty_connect;

$titulo = addslashes($_POST['titulo']);
$desc = $_POST['desc'];
```

Figura 2.18: Código inicial do arquivo inspecao_criar_enviar.php.

As linhas 4 até 10, como já é de nosso conhecimento, incluem arquivos necessários para o desenvolvimento da lógica de inspecao_criar_enviar.php. Da linha 12 até 18, já foi explicado o seu funcionamento. Logo, os comandos que precisam de maior atenção nesta primeira figura estão nas linhas 20 e 21. A linha 20 define uma variável do php chamada titulo e atribui a ela o valor do campo titulo enviado pelo formulário. De maneira semelhante na linha 21, defini-se e atribui-se à variável desc o valor do campo desc. Na Figura 2.19, vemos a segunda parte do código do arquivo inspecao_criar_enviar.php.

```php
if ($titulo == "")
{
    $smarty->assign('form_security_input', FormSecurity::geraParaPost());
    $smarty->assign('titulo_branco', 'erro');
    $smarty->display('inspecao_criar.tpl');
    exit();
}

if ($desc == "")
{
    $smarty->assign('form_security_input', FormSecurity::geraParaPost());
    $smarty->assign('desc_branco', 'erro');
    $smarty->display('inspecao_criar.tpl');
    exit();
}

$grupo = $usuario->inserirGrupo($titulo);

if ($grupo == "ERRO_NOME")
{
    $smarty->assign('form_security_input', FormSecurity::geraParaPost());
    $smarty->assign('nome_erro', 'erro');
    $smarty->display('inspecao_criar.tpl');
    exit();
}
```

Figura 2.19: Continuação do código do arquivo inspecao_criar_enviar.php.

Na Figura 2.19 acontece o tratamento de erros. Pode-se observar que se o usuário não entrar com um título ou a descrição válida, o sistema atribui à variável referente ao erro um valor de string "erro"e então retorna o template inspecao_criar.tpl para que o erro seja analisado e corrigido. Se nenhum erro acontecer, cria-se um grupo que terá no campo nome da tabela o valor referente ao título da inspeção. Se já existir um grupo com título, então o

sistema direciona o usuário novamente à página de criação da inspeção.

Como é mostrado na Figura 2.20, ações para configurar o grupo continuam a ser feitas e uma chamada para configurar a permissão do usuário no grupo também e executada.

```
49 $perfil = 2;
50
51 $grupo = Grupo::getGrupoPorNome($titulo);
52
53 DAOHierarquia::inserir($grupo->getCodigo(), "Usuário Comum", 0, 0, 0, 0, 0, 0, 0, 0);
54
55 $cod_hier = DAOHierarquia::selecionar_maior_codigo($grupo->getCodigo());
56
57 $grupo->definirPerfil($perfil, $cod_hier);
58
59 if ($grupo->getForum() == NULL)
60 {
61     $grupo->criarForum($usuario->getCodigo());
62 }
63 if ($grupo->getMural() == NULL)
64 {
65     $grupo->criarMural($usuario->getCodigo());
66 }
67 if ($grupo->getPortaArquivos() == NULL)
68 {
69     $grupo->criarPortaArquivos($usuario->getCodigo());
70 }
71 if ($grupo->getPortaLinks() == NULL)
72 {
73     $grupo->criarPortaLinks($usuario->getCodigo());
74 }
75
76 //permissoes de perfil
77
78 DAOUsuarioPermissaoPerfil::inserir($usuario->getCodigo(), $grupo->getCodigo(), 1, 1, 1, 1);
```

Figura 2.20: Terceira parte do código do arquivo inspecao_criar_enviar.php.

Até este momento, não foi executada nenhuma ação relativa à inspeção. Na Figura 2.21, pode-se observar chamadas aos métodos DAO da classe DAOInspeção, que é responsável por realizar as consultas no banco referente às inspecões. A criação da classe DAOInspeção será abordada em seguida.

```
80 $insp = DAOInspecao::inserir($grupo->getCodigo(), $desc);
81
82 DAOInspecao::inserir_usuario_inspecao($grupo->getCodigo(),$usuario->getCodigo());
83
84 $smarty->assign("insp",$insp);
85 $smarty->assign('mensagem','Você criou com sucesso a inspeção ');
86
87 $part_insp = DAOInspecao::selecionar_inspecoes_por_usuario($usuario->getCodigo());
88
89 $smarty->assign('insp_dono',$part_insp);
90
91 $smarty->display('inspecao_participa.tpl');
92
93 ?>
```

Figura 2.21: Parte final do código do arquivo inspecao_criar_enviar.php.

Na linha 80 da Figura 2.21, insere-se uma inspeção com o código do grupo que acabou de ser criado e com a descrição que o usuário digitou. Nas linhas seguintes, insere o usuário que criou a inspeção como um avaliador desta inspeção, atribui-se uma mensagem de sucesso para o Smarty, solicita as inspeções que o usuário participa e atribui-as a uma variável de template. Por fim, indica-se ao Smarty que o template resultate desta ação será inspecao_participa.tpl.

A última fase deste desenvolvimento é criar o arquivo DAOInspeção. Cria-se este arquivo no pacote DAO do MISTool, que é onde ficam os arquivos DAO do sistema. A Figura 2.22 ilustra o método montar_objeto , a Figura 2.23 mostra a função inserir e a Figura 2.24 ilustra o método selecionar_por_colunas. A lógica aqui envolvida é simples. O arquivo inspecao_criar_enviar.php solicitou uma chamada à função inserir da classe DAOInspeção. Esta função insere a inspeção na tabela mis_inspecao do banco de dados e solicita uma chamada para o método selecionar_por_colunas passando o código do grupo como parâmetro. Este método consulta o banco, encontra os campos necessários do grupo (inspeção) e faz uma chamada para o método montar_objeto. Por fim, esta função constrói um objeto do tipo Inspeção e retorna-o para o arquivo inspecao_criar_enviar.php.

```php
<?php
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/php/banco/Banco.php");
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/php/dao/DAOGrupo.php");
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/php/dao/DAOUsuario.php");
require_once($_SERVER["DOCUMENT_ROOT"]."/oriongroups/mistool/php/base/Inspecao.php");

class DAOInspecao{

    public static function montar_objeto($array_inspecao){
        if (isset($array_inspecao['cod_grupo'])){
            $insp = new Inspecao();
            $insp->setCodigo($array_inspecao['cod_grupo']);
            $insp->setCodUsuarioDono($array_inspecao['cod_usuario']);
            $insp->setNome($array_inspecao['nome']);
            $insp->setDataCriacao($array_inspecao['data_criacao']);
            $insp->setTipoPerfil($array_inspecao['tipoperfil']);
            $insp->setDescricao($array_inspecao['descricao']);
        }else if (isset($array_inspecao['COD_GRUPO'])){
            $insp = new Inspecao();
            $insp->setCodigo($array_inspecao['COD_GRUPO']);
            $insp->setCodUsuarioDono($array_inspecao['COD_USUARIO']);
            $insp->setNome($array_inspecao['NOME']);
            $insp->setDataCriacao($array_inspecao['DATA_CRIACAO']);
            $insp->setTipoPerfil($array_inspecao['TIPOPERFIL']);
            $insp->setDescricao($array_inspecao['DESC']);
        }
        return $insp;
    }
```

Figura 2.22: Método montar objeto da classe DAOInspeção.

```php
    public static function inserir($id_grupo, $desc)
    {
        $banco = Banco::getBanco();

        $sql = "
                insert into mis_inspecao
                    (cod_grupo, descricao)
                values
                    (
                        '$id_grupo',
                        '$desc'
                        )
        ";
        $banco->executar($sql);
        $grupo_array = DAOInspecao::selecionar_por_colunas(array('cod_grupo'), array($id_grupo));
        return $grupo_array[0];
    }
```

Figura 2.23: Método inserir da classe DAOInspeção.

```php
public static function selecionar_por_colunas($array_colunas, $array_valores){
    $banco = Banco::getBanco();
    $sql = "
            select *
            from
                mis_inspecao,grupo
    ";
    if (count($array_colunas) != 0){
        $sql .= " where ";
        for ($i = 0; $i < count($array_colunas); $i++){
            $sql .= "mis_inspecao.
                    ".$array_colunas[$i]." = '".$array_valores[$i]."'
                    and grupo."
                    .$array_colunas[$i]." = '".$array_valores[$i]."'
                    ";
            if ($i < count($array_colunas)-1){
                $sql .= "
                        and
                ";
            }
        }
    }
    $sucesso = $banco->executar($sql);
    if ($sucesso == FALSE){
        return FALSE;
    }
    $i = 0;
    $inspecoes = array();
    while ($linha = $banco->fetchAssoc()){
        $inspecoes[$i] = DAOInspecao::montar_objeto($linha);
        $i++;}
    return $inspecoes;
}
}
```

Figura 2.24: Método selecionar_por_colunas da classe DAOInspeção.

O restante do desenvolvimento do MISTool procedeu de forma muito semelhante a que foi mostrada nesta seção. Este sistema, apesar de ser relativamente pequeno, é robusto e eficiente devido à boa implementação de seu código. Testes locais foram executados durante toda a sua fase de desenvolvimento e os erros encontrados, solucionados.

2.5 Criação do documento de ajuda

O documento de ajuda (help) do MISTool, foi desenvolvido em um único arquivo que contém tópicos e sub-tópicos que estão linkados com as suas referentes explicações. Aborda-se neste documento de forma prática todo o processo de CEA (Criação, Exclusão e Atualização) de uma inspeção, inserção/exclusão de avaliador em uma inspeção, CEA de signos metalinguísticos, estáticos, dinâmicos e comparação de signos.

Por fim, este documento de ajuda se apresenta de forma clara ao usuário para que este possa ultrapassar quaisquer eventuais problemas no sistema.

2.6 Teste do Sistema

Foram realizados testes de código durante toda a fase de desenvolvimento do MISTool. Os problemas (bugs) encontrados foram solucionados com êxito e o sistema aparenta se comportar

muito bem. Não foram realizados testes com usuários mesmo este sendo uma boa prática para se encontrar erros no sistema.

Capítulo 3

Avaliando Sistemas com o MISTool

O MISTool é um ambiente online colaborativo para a avaliação de sistemas interativos. Esta avaliação poder ser feita utilizando-se um ou mais avaliadores e conforme foi mostrado no Capítulo 1, fundamenta-se nos cinco passos centrais do MIS. Para exemplificar o uso do MISTool, a partir de um usuário fictício chamado Matheus, criou-se uma inspeção Exemplo e inseriu-se um segundo avaliador nesta inspeção conforme as Figuras 3.1, 3.2, 3.3 e 3.4 ilustram.

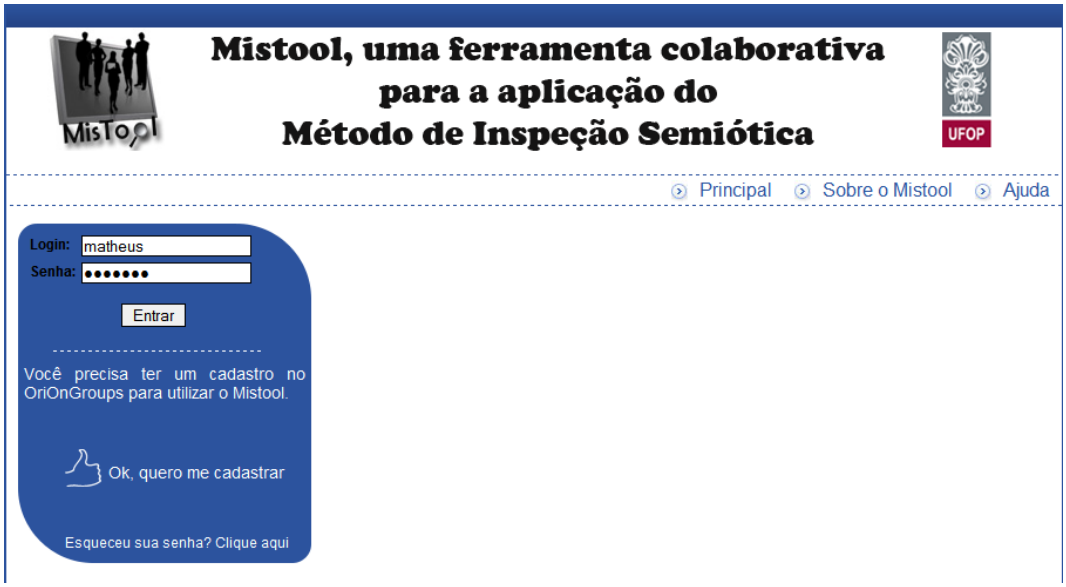

Figura 3.1: Exemplo de um usuário executando um login no MISTool.

Na Figura 3.1 vê-se a tela inicial do MISTool. Um usuário fictício chamado Matheus loga-se no sistema.

Figura 3.2: Exemplo de criação de uma inspeção no MISTool.

Após logar-se no MISTool, o usuário Matheus clica em "Criar nova inspeção". A Figura 3.2 ilustra a criação de uma inspeção no MISTool.

Figura 3.3: Ilustração do estado do sistema após a criação de uma inspeção.

Após criar a inspeção Exemplo, o usuário seleciona-a dentre as inspeções que é dono. A Figura 3.3 mostra a execução desta tarefa.

Com a inspeção selecionada, o usuário insere um novo avaliador na inspeção Exemplo, como pode ser visto na Figura 3.4.

Figura 3.4: Ilustração da inserção de um avaliador em uma inspeção .

Após ter criado a inspeção Exemplo e inserido o avaliador Fabíola, o sistema está pronto para que o usuário inicie a avaliação de signos segundo a aplicação do MIS.

O usuário deste exemplo decide iniciar sua avaliação clicando em "Signos Metalinguísticos", que avalia signos de documentos de ajuda, tutoriais explicativos, termos de serviço e etc.

As Figuras 3.5 e 3.6 ilustram o início e fim da criação de um signo metalinguístico na inspeção Exemplo.

Figura 3.5: Ilustração do início da inserção de um signo metalinguístico.

Figura 3.6: Ilustração do estado do sistema após a inserção de um signo metalinguístico.

A Figura 3.7 ilustra a discussão entre os avaliadores Matheus e Fabíola a respeito do signo inserido.

Após fazer uma avaliação dos signos metalinguísticos, o usuário decide avaliar os signos estáticos. A Figura 3.8 ilustra o início desta operação e a inserção de um novo signo.

Conforme visto na Figura 3.9, o usuário decide inserir o signo "Outra imagem feia"logo após inserir o signo "Imagem do mistool horrível". O resultado desta operação é exibido na Figura 3.10.

Figura 3.7: Ilustração da discussão sobre um signo.

Figura 3.8: Ilustração da inserção de um signo estático.

Figura 3.9: Ilustração da inserção de um novo signo estático.

Figura 3.10: Ilustração do resultado do sistema após a inserção de dois signos.

Neste momento, o usuário decide inserir signos dinâmicos no sistema. Os signos dinâmicos representam muitas vezes interações usuário-sistema e são inseridos na forma de vídeos. A Figura 3.11 ilustra a inserção de um signo dinâmico no MISTool e a Figura 3.12 o estado do sistema após esta inserção.

Figura 3.11: Inserção de um signo dinâmico no MISTool.

Figura 3.12: Resultado do sistema após a inserção de um signo dinâmico no MISTool.

A Figura 3.13 ilustra o resultado do sistema após o avaliador Fabíola ter criado um novo signo dinâmico na inspeção Exemplo. Esta Figura ilustra também como ficam as discussões sobre os signos.

Figura 3.13: Resultado do sistema após a inserção de outro signo dinâmico no MISTool.

Por fim, o último passo na avaliação de um sistema interativo pelo MISTool é fazer a comparação dos signos. Este é o último passo pois Qualidade da Metacomunicação não está disponível nesta primeira versão do sistema.

A Figura 3.14 ilustra a inserção de uma comparação de signo no MISTool. Como pode ser visto também, o usuário pode cadastrar ou excluir perguntas guia no sistema.

Figura 3.14: Inserção de uma comparação de signos no MISTool.

A Figura 3.15 mostra o estado final do sistema após a inserção de uma comparação.

Figura 3.15: Estado do sistema após a inserção de uma comparação de signo no MISTool.

Abordou-se neste capítulo a criação de uma inspeção e a avaliação completa de signos nesta inspeção. O funcionamento geral do MISTool foi visto e também como o produto final deste trabalho pode auxiliar avaliadores que trabalham com o MIS. Com isto, finaliza-se este projeto com a expectativa de que o MISTool agrade a pesquisadores e profissionais da área de IHC.

Capítulo 4

Conclusões

4.1 Introdução

Extendido de um sistema de apoio a comunidades virtuais chamado OriOnGroups, o MISTool foi desenvolvido para suprir uma necessidade na área de IHC, que é obter um sistema robusto apoiado pelo MIS na avaliação de sistemas interativos.

Relativamente pequeno, o MISTool é de fácil manutenção, uma vez que a sua estrutura está dividida em camadas que separam a lógica da aplicação da apresentação.

Com isto, espera-se que o MISTool sendo um sistema inovador, agrade a todos os profissionais da área de Interação Humano-Computador, seja ele um pesquisador, professor, aluno ou do mercado de trabalho.

4.2 Resultados

O resultado obtido é que partindo de um protótipo não funcional, criou-se um produto de qualidade, organizado e com alto desempenho, apto a participar de pesquisas e ser alvo de estudos.

4.3 Trabalhos Futuros

Infelizmente não foi possível desenvolver todas as cinco etapas principais do MIS. A última etapa denominada "Qualidade da Metacomunicação"está ausente nesta versão do MISTool e fica para trabalhos futuros, implementá-la no sistema.

A segunda indicação para trabalhos futuros é realizar testes com usuários, pois no MISTool foram feitos apenas testes de sistema e implementação. A importância de testes com usuários é que muitas vezes estes encontram erros que para desenvolvedor passaram despercebidos.

Capítulo 5

Referências Bibliográficas

Referências Bibliográficas

[1] Norman,D.A.,Draper S.W.(1986). *Cognitive engineering.User Centered System Design. Erlbaum: Hillsdale, NJ.*

[2] da Silva,E.J., de Jesus,A.M.,(2010). *MISTool: um ambiente colaborativo de apoio ao Método de Inspeção Semiótica*

[3] de Souza, C.S., (2005). *The semiotic engineering of humancomputer interaction. Cambridge, MA: The MIT Press,2005.*

[4] da Silva, R.S.,(2006). *Projetando uma Nova Estrutura para uma Comunidade Online. Relatório Técnico. Departamento de Computação. Universidade Federal de Ouro Preto*

[5] de Souza, C.S., Leitão,C.S, Prates,R.O, da Silva,E.J, (2006) *The Semiotic Inspection Method*

[6] de Jesus, A.M., (2006) *Múltiplos Perfis de Usuário em Comunidades Virtuais: Uma Implementação no OriOnGroups. Projeto Orientado. Departamento de Computação. Universidade Federal de Ouro Preto*

[7] *PHP Official Site [online], disponível na Internet via http://www.php.net/, arquivo capturado em 12 de setembro de 2010.*

[8] *Apache Official Site [online], disponível na Internet via http://www.apache.org/, arquivo capturado em 14 de setembro de 2010.*

[9] *MySQL Official Site [online], disponível na Internet via http://www.mysql.com/, arquivo capturado em 14 de setembro de 2010.*

[10] *Eclipse Official Site [online], disponível na Internet via www.eclipse.org/galileo/, arquivo capturado em 25 de agosto de 2010.*

[11] *Smarty Official Site [online], disponível na Internet via www.smarty.net, arquivo capturado em 01 de setembro de 2010.*

[12] Carrol,J.M.,(2003) *HCI Models, Theories and Frameworks: Toward a multidisciplinary science. San Francisco Morgan Kaufmann Publishers*

[13] Carrol,J.M.,(2000) *Making Use: Scenario-Based Design of Human-Computer Interactions, MIT Press, Cambridge, MA.*

[14] Silveira, M.S.,(2002) *Metacomunicação Designer-Usuário na Interação Humano-Computador. Tese de Doutorado. Departamento de Informática.PUC-RIO*

[15] Woods D.D.,(1994) *Observations from studying cognitive systems in context. In Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society (Keynote address)*

[16] Dowell,J., Long,J.,(1998) *Conception of the Cognitive Engineering design problem.Ergonomics, vol.41, 126-139*