Ferramenta de Apoio aos Testes de Software de Emissão Fiscal PAF-ECF

Alan Barbosa da Silva, Edmundo Sérgio Spoto
UFG - INF, 74.001-090, Brasil
alanbarbosa.gyn@gmail.com e dinospoto@gmail.com

PALAVRAS-CHAVE: Desenvolvimento de software, Teste Funcional de Software, Homologação de Software, PAF-ECF.

1 INTRODUÇÃO

A partir de 2008 o Conselho Fazendário do Brasil (CONFAZ) tomou uma decisão de obrigar as empresas desenvolvedoras de software a passar por uma avaliação em seus softwares seguindo Normas, Resoluções e Convênios editados pela Cotepe (Comissão Técnica Permante do ICMS) no Ato Cotepe/ICMS 15/2008 credencia Orgãos Técnicos para a realização de Análise Funcional de Programa Aplicativo Fiscal (PAF-ECF). No Convênio ICMS 15/2008 de 4 de abril são apresentados regulamentações da prática em que os órgãos técnicos devem se submeter para a realização dos testes seguindo o roteiro de Análise Funcional de Programas de Aplicativo Fiscal e Emissão de Cupom Fiscal (PAF-ECF).

A Universidade Federal de Goiás, junto com o Instituto de Informática se tornou um Orgão Técnico credenciado para realizar as Avaliações, e a partir de 2010 deu ínicio aos trabalhos no Laboratório do Centro de Tecnologia de Software. Dentro das atividades de realização da Homologação de um software que utiliza PAF-ECF, existem várias atividades que são desempenhadas ao longo da Homologação. Sendo assim existiu uma necessidade de se organizar as informações que são geradas para essa finalidade.

O projeto de Validação e Verificação de Software visa estabelecer um processo de técnicas de Teste com base no Roteiro de Análise Funcional PAF-ECF emitido pela COTEPE/ICMS ao longo do período de atividades. Já se passaram por várias versões do Roteiro e a cada período os testes são

especializados e melhorados visando tornar as atividades de testes cada vez mais segura e mais efetiva visando detectar se o software em teste cumpre com os requisitos básicos descritos nos atos e convênios emitidos pela COTEPE.

Este trabalho de iniciação científica se propos a desenvolver uma ferramenta de apoio ao gerenciamento das atividades de testes, podendo assim ter melhores resultados e feedbacks ao longo do período de testes que um Roteiro de Análise Funcional permanece ativo, registrando assim os tipos de erros encontrados, quem trabalhou nas atividades de testes, os softwares que passaram pelos testes, as principais modificações efetuadas e o período de execução de cada homologação.

Segundo o Convênio COTEPE/ICMS 15/08 é fundamental que todo software ao iniciar sua avaliação seja gerada o MD5, e no final do teste o MD5 será gerado novamente, mostrando assim que o software passou por todos os requisitos e testes sem passar por modificações ao longo do teste. É muito comum ao longo dos testes a ocorrência de intervenções do programador das empresas nos seus softwares efetuando modificações e alterações. Para isso é necessário um processo efetivo visando atender o convênio 15/08, pois existem modificações que podem afetar partes do software que já foram testadas anteriormente, obrigando assim a repetição desses testes.

Este trabalho teve como motivação preparar uma ferramenta de apoio para gerenciar as atividades de testes de um determinado software e poder armazenar todas as intervenções (alterações) que foram geradas pela empresa ao longo do período de teste, o número de vezes que o software teve que passar pelos mesmos testes devido às modificações e observar resultados importantes para a pesquisa em relação a eficácia dos testes realizados e quais testes que mais detecta defeitos nos softwares.

Segundo MEYERS (2004) o teste é a única maneira de saber se um software possui defeitos, com a sua execução. A geração de casos de testes depende dos critérios adotados e da observação do testador em relação aos requisitos envolvidos. Por se tratar de testes caixa preta (Baseado na Funcionalidade) o Roteiro não obriga que um determinado dado de teste seje gerado para atender um determinado teste, porém ele aprimora a visão do escopo a ser testado visando assim levar a geração de determinados dados de

entrada e apresenta os possíveis resultados esperados no teste, obrigando assim a avaliação de não atendimento ao requisito caso os passos apresentados em cada etapa do teste não seja atendido.

Neste trabalho na Seção 2 são apresentadas as técnicas e definições dos termos utilizados na confecção da ferramenta de apoio ao teste funcional de Programas PAF-ECF. Na Seção 3 são apresentados os diagramas de Entidade e Relacionamento do Banco de Dados e os Diagramas de Projeto em UML. Na Seção 4 é apresentado um estudo de caso do funcionamento da ferramenta atribuindo assim as principais funcionalidades que ela oferece. Na Seção 5 são apresentados os Resultados Finais e Trabalhos Futuros.

2 Definições e Terminologias

Nesta seção pretende-se descrever as principais definições e terminologias utilizadas em cada parte das técnicas abordadas para a confecção da ferramenta de apoio ao teste funcional PAF-ECF. A seguir serão apresentados os principais termos e definições da parte de Banco de Dados.

2.1 Banco de Dados

A geração de um Banco de Dados é fundamental para que as informações geradas e consumidas por um determinado software seja armazenada com segurança e recuperada em tempo de execução. Segundo SILBERCHATZ et al (1999), DATE (2000) e EMASRI e NAVATHE (2002) um Sistema Gerenciador de Banco de Dados contribui para que um projeto de Banco de Dados seja construído com sucesso quando as etapas de requisitos, projeto lógico e projeto físico estiverem prontos. A seguir os códigos de geração das tabelas e metados (restrições requeridas pelo projeto) são gerados e passado para o SGBD. Na Seção 3 serão apresentados os diagramas de entidade e relacionamento DER e o diagrama de Classe referente à construção da Ferramenta.

A construção do Diagrama de Entidade e Relacionamento é o ínicio da elaboração do Banco de Dados em um dado SGBD. Ele deve conter os atributos de cada tabela, os relacionamentos descritos de forma correta

indicando relacionamentos de dependências funcionais entre as tabelas e também as chaves que irão identificar cada entidade. No projeto físico é necessário que cada atributo esteja definido pelo seu respectivo tipo de dado que melhor represente a sua funcionalidade no sistema.

Neste trabalho foi adotado o SGBD Oracle e PostGreeSQL por existir compatibilidade na formação das ações em que o sistema de banco de dados deverá conter. Acredita-se que o PostGreeSQL por ser um SGBD Open Source pode representar todas as necessidades que estão sendo colocadas para este propósito.

O SGBD pode ser definido como um software com recursos específicos para facilitar a manipulação das informações armazenadas no Banco de Dados. Segundo DATE (2000) o SGBD surgem para resolver problemas que os sistemas tradicionais de arquivos possuiam como redundância de dados, inconsistência e a dificuldade de controle de acesso. Conforme (SILBERCHATZ, 1999) o objetivo de um SGBD é proporcionar um ambiente tanto conveniente quanto eficiente para a recuperação e armazenamento das informações do banco de dados.

Atualmente existem diversos SGBDs disponíveis comercialmente sendo utilizados em organizações ou instituições de pesquisa. Cada um possui tecnologias diferenciadas que podem contribuir com as necessidades específicas de cada projeto. Considera-se fundamental em um SGBD a segurança da informação, conforto de recuperação e manipulação e o controle das integridades (semânticas, referenciais, de chave, de tabela, dependência funcional e outras).

Segundo PRESSMAM (2004) todo projeto de software deve passar por etapas que levamtam inicialmente os requisitos básicos e necessários que um software pretende atender. Sendo assim para que a construção da ferramenta ocorra foi necessário passar por uma breve análise dos requisitos a serem alcançados extraídos do Roteiro de Análise Funcional PAF-ECF, além do levantamento das funcionalidades principais e das entidades envolvidas em cada atividade do Teste.

Na próxima seção são apresentados os conceitos de Casos de Usos e Classes que serão utilizados como parte do projeto para a composição da ferramenta de apoio ao teste funcional PAF-ECF.

2.2. UML – Unified Modelling Language

UML é uma linguagem para visualização, especificação, construção e documentação de artefatos de um software em desenvolvimento.

A UML é composta por vários diagramas que permitem modelar, analisar e descrever as principais etapas do projeto de um sistema. Neste trabalho foram usados apenas as etapas de requisitos (diagrama de casos de usos) e as etapas de construção do sistema no diagrama de Classe.

O Diagrama de Casos de Usos tem como objetivo auxiliar a comunicação entre os analistas e o cliente. Um Diagrama de Casos de Usos descreve um cenário que mostra cada funcionalidade do sistema do ponto de vista do usuário (Ator). Cada usuário no Diagrama de Caso de Uso representado pelos atores que irão interagir com as funcionalidades que o sistema se propõe a realizar.

Para a representação das funcionalidades são utilizadas um simbolo oval com o nome da funcionalidade, esse simbolo é denominado de caso de uso. O ator ou usuário do sistema é representado por um simbolo que é o desenho de um homem, mas podendo ser tambem um sistema (como um sistema acadêmico ou sistema financeiro e outros).

Na Figura 1 é apresentado um modelo do Diagrama de Casos de Usos. A seta representa que o Ator (usuário) interage com o Caso de Uso.

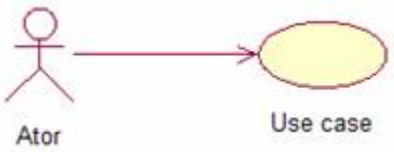

Figura 1: representação de um Diagrama de Caso de Uso

O Diagrama de classe descreve os vários tipos de objetos no sistema e o relacionamento entre eles. O Diagrama de classe é construído inicialmente sob a perspectiva dos conceitos atribuídos na análise de requisito, onde são levantados apenas as classes abstratas e seus respectivos atributos. Em seguida são construídos diagramas de sequencias que levantam quais objetos se comunicam com mensagens com outros objetos onde são levantados os métodos nas classes destinos do diagrama, sendo assim o

diagrama de classe passa a ser construído levando em consideração as ações que cada objeto deverá exercer no sistema quais métodos são responsáveis por cada objeto. Na Figura 2 é apresentado o diagrama de classe mostrando como cada classe representada por um retangulo contendo: nome, atributos e métodos. Podendo, porém existir classes só com atributos ou só com métodos dependendo da sua composição.

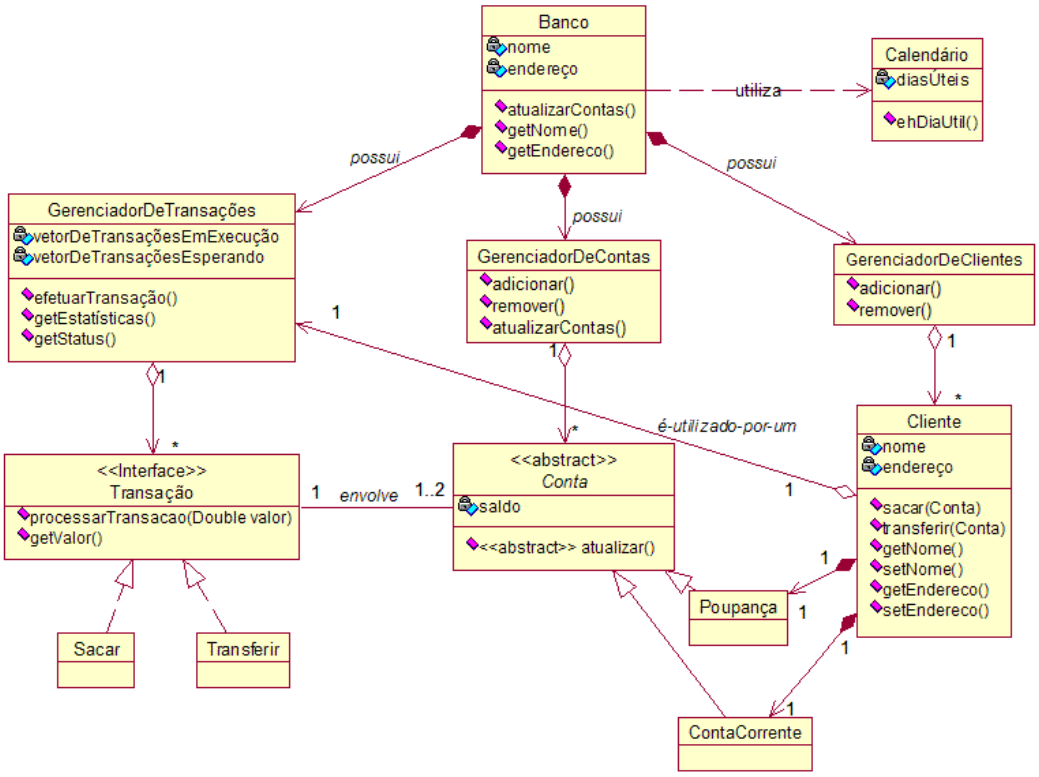

Figura 2: Exemplo de um Diagrama de Classe (UML)

2.3. PHP – Hypertext Preprocessor

Criado por Rasmus Lerdorf para rastrear os usuários em seu site, tornou-se uma das linguagens de script mais populares para a criação de páginas Web dinâmicas no lado do servidor.

Segundo manual da PHP(2011),

> "*A nova engine, dublado 'Zend Engine' (conhecidos pelos seus primeiros nomes, Zeev e Andi), fazendo desse objetivo um sucesso, e foi introduzida em meados de 1999. PHP 4.0, baseado nesta engine, e acompanhado com uma série de novas*

> *características, foi oficialmente lançada em Maio de 2000, quase dois anos após o seu predecessor, o PHP 3.0. Além da altíssimo melhoramento da performance desta versão, o PHP 4.0 incluiu outras características chave como o suporte para muitos servidores WEb, sessões HTTP, buffer de saída, maneiras mais seguras de manipular input de usuários e muitas construções novas na linguagem*".

O PHP é uma tecnologia de código aberto que conta com o apoio de uma grande comunidade de usuários e desenvolvedores. É independente de plataforma, havendo implementações para todos os principais sistemas operacionais, como UNIX, Linux, Mac e Windows, e também aceita muitos bancos de dados.

Segundo Deitel(2008) A interatividade entre usuário e servidor tornou-se parte fundamental da funcionalidade da Web, tornando o PHP – linguagem escrita especificamente para interagir com a Web – uma ferramenta valiosa.

No paradigma Orientado à Objetos (OO), trabalha-se com o conceito de objetos que se comunicam apartir do envio de mensagens. Uma mensagem é um pequeno texto que os objetos entendem, através desta solicita e também envia informação (parâmetros). Outros conceitos como herança e polimorfismo utilizados nesse paradigma ajuda o programador a reutilizar código e desacoplar o software desenvolvido, faciltando a manuntenção e a posterior reutilização de classes do mesmo.

3 FERRAMENTA DE APOIO AO TESTE

A Ferramenta de Apoio ao Teste de Software é composta pela arquitetura apresentada na Figura 3.

Baseada no modelo Cliente-Servidor, em três camadas, Cliente, Aplicação, Banco de dados. O lado Cliente é responsável pela exibição e a entrada de dados. As informações são geradas pelos homologadores, durante a execução dos testes, pela empresa mediante o preenchimento da ficha de inscrição do laudo, e pelo administrador durante a finalização da análise.

O lado cliente faz solicitações ao servidor da aplicação, que se necessário for, consulta o banco de dados, o que vai acontecer praticamente sempre, o banco por sua vez retorna os dados solicitados pela aplicação, que responde a requisição com os dados em uma página web (HTML ou PHP). E como produto final principal é gerado o laudo, que é um relato dos resultados obtidos seguindo o Roteiro de Teste da COTEPE/ICMS.

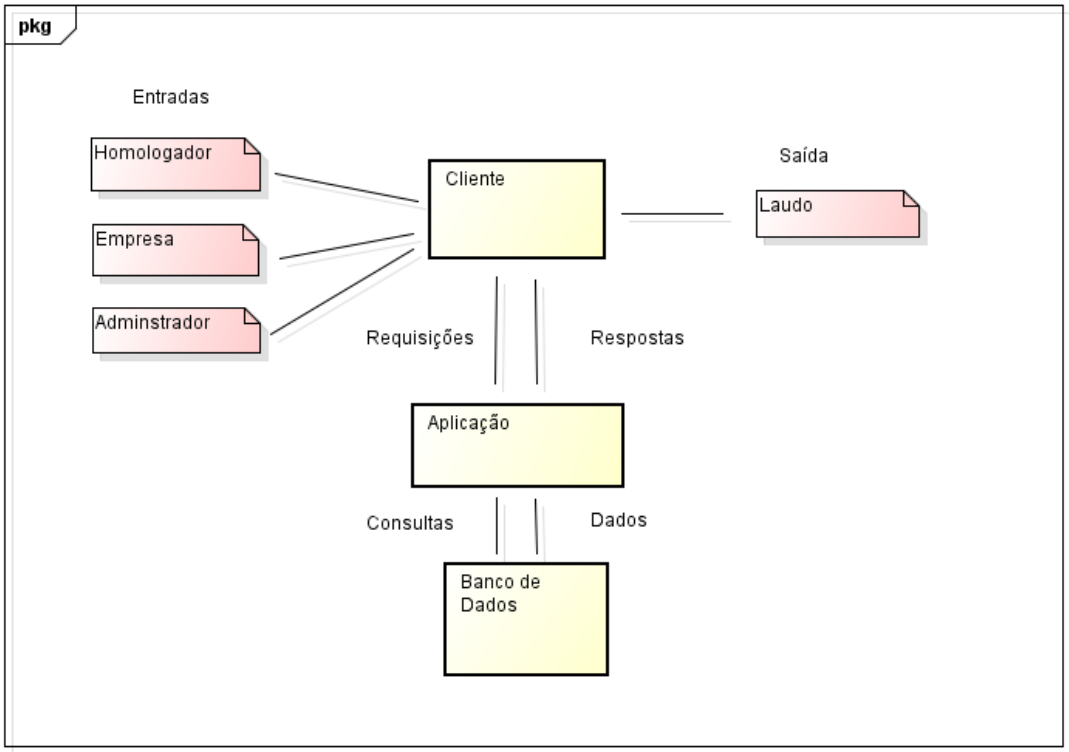

Figura 3: Arquitetura da Ferramenta de Apoio ao Teste Funcional PAF-ECF

3.1 – Diagrama de Caso de Uso

O Diagrama de Caso de Uso da Ferramenta de Apoio ao Teste Funcional PAF-ECF é mostrada na Figura 4. Os atores representam os principais usuários do sistema que são responsáveis pela geração das informações e interação com as funcionalidades (casos de usos).

A Ferramenta apoia o processo de homologação, que contem as etapas de inscrição do laudo, através do preenchimento de uma ficha, contendo dados do software e da empresa, essa etapa representa o caso de uso Preencher Ficha Inscrição, que é papel da empresa, ator no diagrama. Essa ficha é fundamental para todo o processo, pois através dos dados contidos nela que a análise tomará início, contribuindo também para a geração

do laudo no fim do processo.

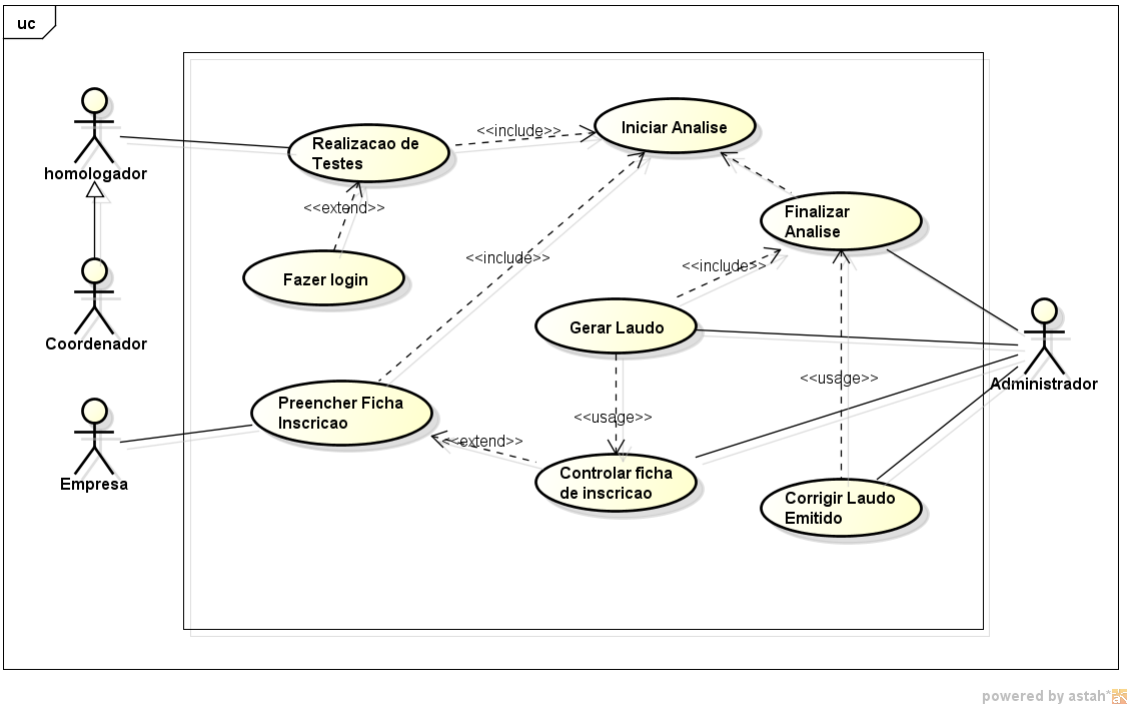

Figura 4: Diagrama de Caso de Uso da Ferramenta

O homologador é responsável pela execução do caso de uso Realização de Testes, sendo que para isso ele deve estar logado. O Administrador é encarregado do caso de uso Controlar ficha de inscrição, Gerar laudo e Finalizar Análise. A geração do laudo será feita após a realização de todos os testes relativos ao tipo de PAF-ECF, e com as informações de duas chaves, MD5 do principal executável e MD5 do conjunto de arquivos executáveis, que são obtidas no final da análise. Obtida as chaves e verificada a consistência da ficha de inscrição, o laudo pode ser gerado com as informações da análise.

O caso de uso Corrigir Laudo Emitido é executado por um administrador da ferramenta foi percebido quando nos deparamos com a necessidade de refazer laudos por falta de informações, dados preenchidos incorretamente ou por exigência de órgãos de alguns estados. Nesse caso de uso trabalhasse a atualização de dados, revisão e emissão de outro laudo, com base na mesma análise do PAF-ECF, alterando somente os dados incorretos e observações.

3.2 – Diagrama de Classe

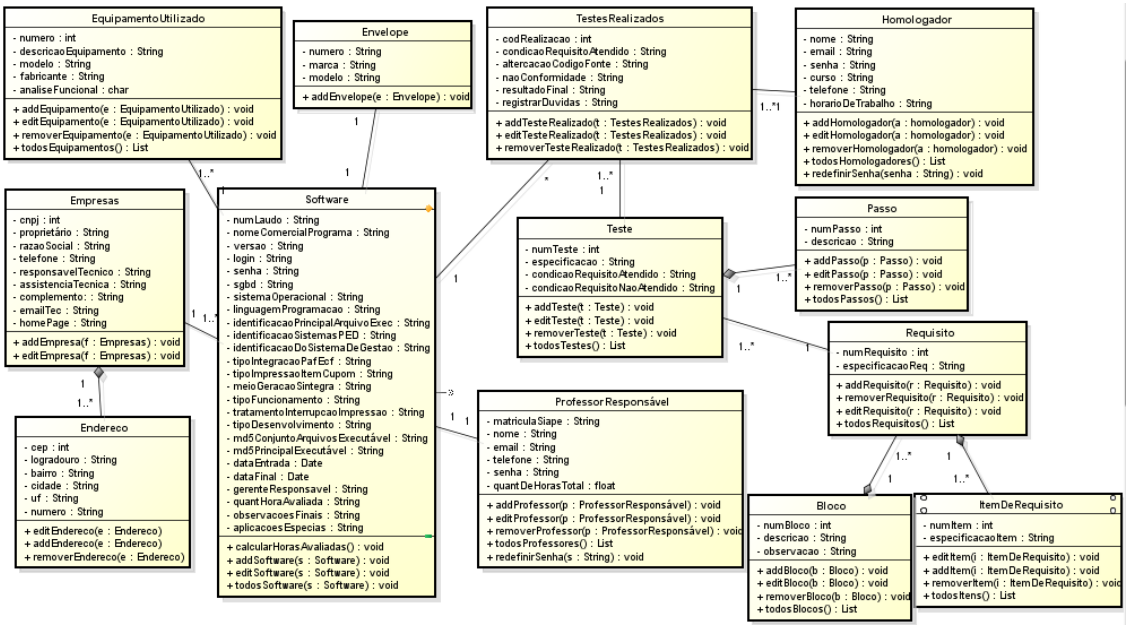

Figura 5: Diagrama de Classe da Ferramenta

O Diagrama de Classe da ferramenta é representado pela Figura 5 onde as classes representam moldes dos objetos que serão instânciados pela ferramenta durante sua execução. Pela imagem pode-se perceber que a classe software é uma das mais importantes, pela quantidade de informações que os objetos instaciados a partir dela guardarão. A análise do software será realizada a partir dos dados da realização dos testes, os quais serão armazenados pelos objetos da classe *TesteRealizados*, dados essenciais para geração do laudo final.

As Classes Professor e homologador serão usadas para o controle de acesso e administração das atividades realizadas durante a análise. Algumas alterações serão necessárias durante a execução do teste, sempre que houver resultados que não satisfaça as condições do roteiro. Neste caso é avaliado o tipo de alteração podendo obrigar a realização dos testes ja executados novamente, e estas informações serão armazenadas para posteriormente serem análisadas.

3.3 – Diagrama de Entidade e Relacionamento (DER)

O DER é apresentado na Figura 6 e representa cada tabela utilizada para o controle das informações que a ferramenta irá trabalhar. Existe uma tabela que controlará as pessoas envolvidas no teste que serão os membros do grupo técnico de teste. A Tabela Empresa armazena as informações da empresa que produziu o software a ser homologado. Na Tabela Software armazena as informações necessárias do software exigido no ato COTEPE.

As tabelas de requisitos, passos, blocos e testes são as informações extraídas do Roteiro de Análise Funcional PAF-ECF e as demais tabelas controlam as atividades de execução do teste durante a homologação.

O DER abaixo guarda todas as informações pertinentes do processo de homologação, desde a solicitação da empresa para análise do PAF-ECF, até a geração de laudo. Registrando informações importantes como tempo gasto na realização de um teste, qual homologador que o realizou, quem estava gerenciando a análise, o professor, representado pelo ator coordenador que é responsável pela homologação, e quais as inconformidades encontradas durante os testes.

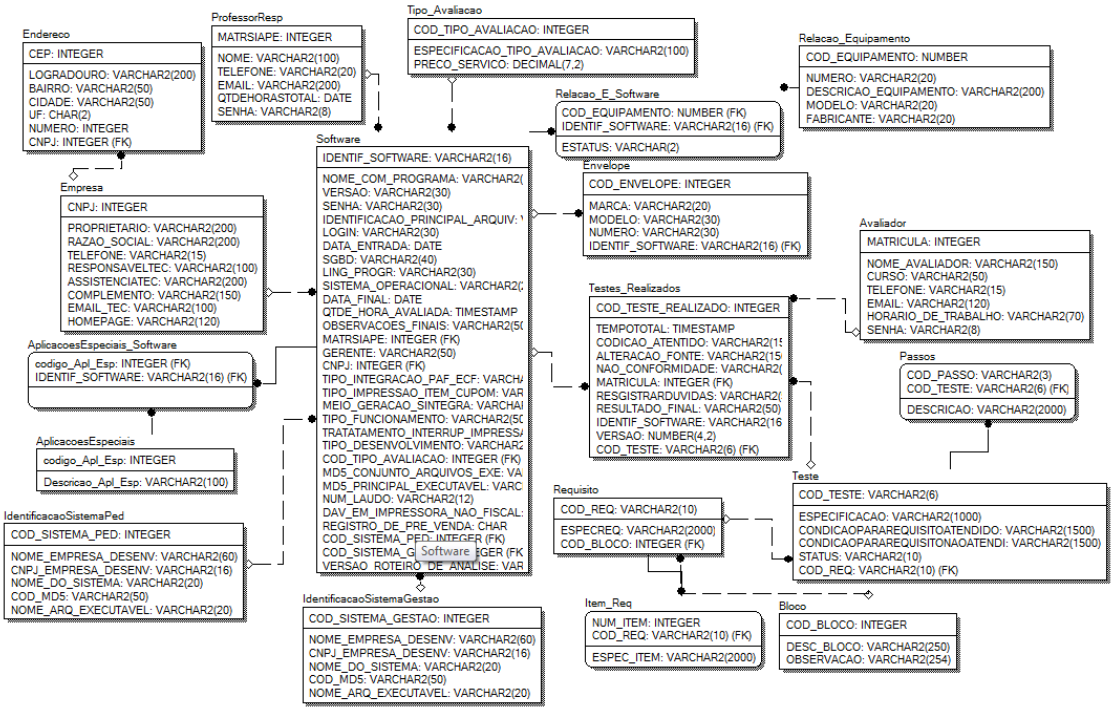

Figura 6. Diagrama Entidade Relacionamento

Além de dados que caracterizam o PAF-ECF, como forma de desenvolvimento, linguagem de programação utilizada, ECF usuado na análise, lista de ECFs que são compatíveis com o sistema, dados do sistema gestão que funcionam integrados ao PAF-ECF, entre outros dados.

4 RESULTADOS OBTIDOS

Com a geração do banco de dados e as principais funcionalidades apresentadas na Seção Anterior foram inseridas as informações referentes aos laudos homologados no PAF-ECF da UFG no período de janeiro a maio de 2011. Esta etapa se encontra em desenvolvimento previsto no cronograma e seu encerramento será em junho de 2011.

Com as alterações que vem ocorrendo no Roteiro de Análise Funcional PAF-ECF alguns ajustes nas informações armazenadas na ferramenta no inicio do ano levaram a corrigir os blocos que foram adicionados. Durante a fase de desenvolvimento do banco e do projeto, conseguimos compreender as mudanças ocorridas no roteiro, com isso buscamos criar uma estrutura para o banco de dados, para ser capaz de sofrer alterações sem grandes impactos nos dados já persistidos.

A partir de Abril de 2011 um novo roteiro versão 1.5 aplicavel à versão 01.07 da ER-PAF-ECF (exceto os requisitos XXXV XXXVI).

5 RESULTADO FINAL E TRABALHOS FUTUROS

Com a realização da Ferramenta para apoiar as etapas de testes funcionais dos Programas de Apoio Fiscal e de Emissão de Cupom Fiscal no Laboratório de Tecnologia de Software da UFG, tem contribuído com a principal situação que até o presente momento vinha ocorrendo, a perda das informações de cada conjunto de testes realizados em uma homologação de software. Até o presente momento já foram realizados 32 Laudos sendo que apenas 24 softwares é que foram avaliados, 8 Laudos foram gerados visando corrigir erros de escritas ou falhas na sua composição.

Espera-se que com a ferramenta a maior contribuição será na confecção das informações e recuperação (feedback) dos resultados obtidos.

As correções de ações de erros e também informações de quais requisitos de testes são mais efetivos a detecção de defeitos nos software.

As operações de povoar o banco de dados com as informações do roteiro e também as informações referentes aos laudos ja realizados foram executados nesta etapa e pretende-se encerrar até o final do mês de junho. Em seguida pretende-se iniciar as etapas de realização de testes quando a ferramenta estiver toda testada em relação às integridades e seguranças das informações nela inseridas.

Como trabalho futuro pretende-se criar um sistema de informação para contribuir com as análises de resultados e estatísticas visando gerar relatórios que poderão contribuir nas decisões dos tipos de requisitos e processo de testes existentes. Também como próximo passo será gerado um módulo que contribuirá com a geração de um processo de testes em relação a auxiliar na geração dos dados de forma efetiva, visando assim tornar o processo de teste mais uniforme e independente do testador envolvido.

6 BIBLIOGRAFIA

DATE,C.J. Introdução a Sistemas de Bancos de Dados, Editora Campus Ltda, 2000.

DEITEL, P.J., DEITEL, H.M. Ajax, Rich Internet Applications e Desenvolvimento Web para programadores, Ed. Pearson Prentice Hall, 2008.

ELMASRI,R and NAVATHE, S.B. Sistemas de Banco de Dados – Fundamentos e Aplicações, LTC Editora, 2002.

LARMAN, C. Utilizando UML e Padrões, Bookman, 2007.

SILBERSCHATZ,A.; KORTH,H.F.; SUDARSHAN,S. Sistemas de Banco de Dados, 3a. edição, Makron Books, 1999.

PHP, disponível em < http://php.net/manual/pt_BR/history.php.php >, acessado em 10/05/2011.