UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS

UNIDADE ACADÊMICA DE EDUCAÇÃO CONTINUADA

ESPECIALIZAÇÃO EM QUALIDADE DE SOFTWARE

Martin Günter Borchardt Bittencourt

DESENVOLVIMENTO DE PROCESSO DE VALIDAÇÃO DE FIRMWARE DE UM ANALISADOR DE ENERGIA

São Leopoldo

2015

UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS

UNIDADE ACADÊMICA DE EDUCAÇÃO CONTINUADA

ESPECIALIZAÇÃO EM QUALIDADE DE SOFTWARE

Martin Günter Borchardt Bittencourt

DESENVOLVIMENTO E IMPLANTAÇÃO DE PROCESSO DE VALIDAÇÃO DE FIRMWARE DE UM ANALISADOR DE ENERGIA

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Especialista em Qualidade de Software, pelo curso de Pós-Graduação Lato Sensu em Qualidade de Software da Universidade do Vale do Rio dos Sinos – UNISINOS.
Orientador: Prof. Dr Regina Maria Thienne Colombo

São Leopoldo

2015

Desenvolvimento e Implantação de Processo de Validação de Firmware de um Analisador de Energia

Martin Günter Borchardt Bittencourt[1]

[1]Unidade Acadêmica de Educação Continuada – Universidade do Vale do Rio dos Sinos (UNISINOS) – São Leopoldo – RS – Brazil

martin_bittencourt@hotmail.com

Abstract. *In the current scenario of the software industry, increasingly companies invest in improving the quality of their products in order to obtain better technical and commercial results. Part of these efforts focus on the establishment of developmental processes, one of these being the validation process that allows to conclude, by applying an organized series of tests, if the created product is one that is intended to get under project's scope. This article presents the development of a validation process in accordance to the CMMI-Dev 1.3 model validation process for the user interface firmware of a power quality analyzer. Not only the process received full approval, but this work also culminated in the creation of a new test case management tool.*

Resumo. *No cenário atual da indústria de software, cada vez mais empresas investem na melhoria da qualidade de seus produtos com o fim de obter melhores resultados técnicos e comerciais. Parte destes esforços concentram-se no estabelecimento de processos de desenvolvimento, um destes sendo o processo de validação que permite concluir, por meio da aplicação de uma série organizada de testes, se o produto criado é aquele que se pretendia obter segundo o escopo de um projeto. Este artigo apresenta o desenvolvimento de um processo de validação segundo o processo de validação do modelo CMMI-Dev 1.3 para o firmware de interface do usuário de um analisador de qualidade de energia. Além do processo ter recebido plena aprovação, este trabalho culminou na criação de uma nova ferramenta de gerenciamento de casos de teste.*

1. Introdução

Assim como a aviação, o surgimento do mundo virtual com o desenvolvimento do computador digital foi uma dos maiores eventos do século XX. Consistindo numa sequência de instruções capazes de comandar o comportamento de um hardware [Niquette, 1995], o objeto virtual denominado "software" passou a dominar o espaço do mundo moderno, sobretudo após a popularização dos computadores pessoais na década de 80. Cada vez mais tarefas, dos mais variados tipos, têm sido delegadas a computadores e seus softwares, um fenômeno que tem sido acompanhado pelo crescimento do número de desenvolvedores e gestores de tais ferramentas.

Assim, como qualquer outro produto criado por mecanismos e trabalhadores falhos, softwares acabam por serem sujeitos à possibilidade de conterem erros de construção sujeitando-os a executar tarefas indesejadas ou a deixar de executar as tarefas para eles planejadas. Esse risco de defeito é entendido como um problema de engenharia e de *qualidade de software* e tem recebido crescente atenção da indústria desde o fim da década de 60 [Wirth, 2008] em meio à "*Software Crisis*" [Brennecke e

Keil-Slawik, 1996] quando o ramo denominado Engenharia de Software começou a ser desenvolvido em meio às obras pioneiras de Boehm [Boehm, 1981], Myers [Myers, 1979] entre outros, já tendo se estabelecido na década de 80 como uma "disciplina em seu próprio direito" [Sommerville, 1982].

A preocupação com a qualidade possui uma história de pouco menos de um século [Pressman, 1995], embora já tenha passado por vários estágios de desenvolvimento [Garvin, 2002] e conte atualmente com uma literatura vasta a seu respeito. Todos esses trabalhos giram em torno do entendimento do que é "qualidade" e da busca a esta, não havendo uma única definição universal para esse termo. Por exemplo, o PMBOK define-a como "o grau com que um conjunto de características inerentes atende aos requisitos" [PMBOK, 2008] enquanto a IEEE STD. 610-1990 define "qualidade" como "o grau que um sistema, componente ou processo cumpre com os requisitos especificados e as necessidades ou expectativas do cliente" [IEEE, 1990], mas ainda outras definições são apresentadas em [Pressman, 2001], [Bartié, 2002] entre outros. Independente da maneira como for definida, entende-se que a busca pela qualidade é a busca pela obtenção de um produto que esteja de acordo com o que se espera dele e para ele.

Com a busca pela qualidade em software, muitas técnicas, métodos, procedimentos e modelos têm sido propostos tanto pela indústria e pela comunidade acadêmica quanto por órgãos governamentais e não governamentais. Para este trabalho, três são os itens que se destacam: o modelo de referência *Capability Maturity Model – Integration for Development* (CMMi-Dev) [CMMI, 2010], processos de desenvolvimento e testes.

O CMMi-Dev, atualmente na versão 1.3, é um modelo de referência contendo práticas para a área de desenvolvimento de software visando o aumento da maturidade das empresas em suas práticas de desenvolvimento e manutenção de produtos de software [CMMI, 2010] [Galagarra, 2014]. Voltado tanto para engenharia de sistemas quanto engenharia de software, ele fornece uma maneira aprimorada de como uma empresa pode organizar e realizar várias das tarefas executadas no decorrer do curso de vida de um projeto. O CMMi-Dev é organizado numa estrutura que começa com cinco grupos chamados "níveis de maturidade", do mais básico ao mais avançado. Cada um desses níveis de maturidade é por sua vez dividido em "áreas de processo" que são as áreas de desenvolvimento da empresa. Uma vez que tal atividade tenha sido realizada, a expectativa, confirmada pela experiência do mercado, é que a empresa se tornará mais eficiente e madura no desenvolvimento de seus produtos que acabarão porventura a ter maior qualidade.

Outra maneira de aumentar a qualidade de produtos desenvolvidos é estabelecer processos de desenvolvimento. Por "processos" entende-se um conjunto de ações organizadas e voltadas a produzir um resultado específico [Becker, 2014]. A presença de tais processos nas empresas é um dos grandes contrapontos aos grupos de trabalhos anteriores à Revolução Industrial compostos por artesãos que criavam os seus produtos cada um à sua maneira. O fruto daquele sistema eram resultados de qualidade variável, sem padronização, e que facilmente não atendiam aos anseios dos clientes. Com o crescimento do número de produtos compostos por várias partes vindas das mais variadas fontes e o aumento dos parques fabris empregando diversos funcionários trabalhando conjuntamente para produzir cada vez mais produtos, tal variabilidade

passou a ser inadmissível forçando as empresas a adotarem mecanismos que garantissem uma maior padronização de seus produtos. Foi nesse contexto que a adoção de processos estabelecidos ao longo da cadeia produtiva se mostrou eficaz na obtenção de melhores resultados.

Embora a adoção de processos tenha chegado mais tarde à indústria de software após o surgimento da Engenharia de Software [Pressman, 1995], a capacidade desta estratégia para melhorar a qualidade de produtos permaneceu a mesma. Atualmente apenas parte dos esforços empregados em melhoria da qualidade de produtos é direcionada aos próprios enquanto outras são voltadas tanto para implantação quanto aprimoramento de processos de desenvolvimento existentes.

Por fim, tem-se a prática de testar o software. Partindo-se da premissa que por melhores que sejam os processos e por melhores que sejam as equipes de desenvolvedores, erros ainda estarão presentes no produto criado [Beizer, 1990], testes acabam por ser indispensáveis tanto na busca por um produto que atenda às expectativas dos clientes quanto na melhora da imagem da empresa. Estes, segundo o que tem sido apresentado na literatura [Delamaro *et. al.,* 2007] [Molinari, 2008] [Naik e Tripathy, 2008], podem ser categorizados de diversas maneiras na medida em que contemplam os diversos aspectos de um software tais como sua segurança, comunicação com outros aplicativos, solidez dos dados, etc.. Testes também têm sido divididos quanto ao objetivo com o qual são realizados: se para confirmar se o produto foi desenvolvido corretamente ou se o produto correto foi desenvolvido. Este segundo grupo tem sido chamado de "testes de validação" [Pressman, 1995] [Sommerville, 2011] e são mais comumente realizados nos estágios finais de construção do software ou de suas partes do que os testes do primeiro grupo.

Em acordo com o que foi mencionado até então, a empresa gaúcha Embrasul Indústria Eletrônica Ltda. optou pela implantação de um processo de validação para o seu mais recente produto, o Analisador de Qualidade de Energia RE8000. O RE8000 é o primeiro analisador brasileiro a cumprir integralmente a Norma Internacional IEC 61000-4-30 Classe A e possui uma tela *touch screen* de sete polegadas de onde pode ser inteiramente controlado. Ele possui uma série de funcionalidades tais como visualização em tempo real de mais de 15 grandezas físicas medidas ou calculadas, registro de medição para posterior análise num computador e diversas possibilidades de configuração. A parte virtual consiste de três firmwares: um código em VHDL para o processamento de dados; um aplicativo centralizador escrito na linguagem C denominado "Center"; e a interface do usuário desenvolvida em C++ com a *framework* de desenvolvimento Qt, da Digia, que tange todas as funcionalidades do aparelho. Estes três firmwares compõe um sistema complexo e consequentemente susceptível à presença de vários erros de programação justificando a criação de um rigoroso sistema de testes para garantir que o produto irá funcionar como foi planejado.

O presente artigo apresenta o trabalho realizado na Embrasul Indústria Eletrônica Ltda. que teve por objetivo principal o desenvolvimento de um processo de validação para o Analisador RE8000 seguindo as boas práticas já estabelecidas no mercado. Como objetivos secundários, o trabalho busca fazer deste o processo modelo para os demais projetos da empresa, conseguir reforçar a cultura de busca pela qualidade no setor de Pesquisa e Desenvolvimento da empresa e obter maior satisfação de seus clientes.

Este artigo está organizado da seguinte forma: na segunda parte, intitulada "Referencial Teórico", constam os fundamentos teóricos utilizados na implementação do processo de validação. Estes fundamentos incluem não apenas uma revisão da literatura relevante quanto aos principais temas abordados como também novos desenvolvimentos teóricos frutos das reflexões surgidas ao longo do desenvolvimento deste trabalho. Na terceira parte, denominada "Metodologia", consta uma descrição dos trabalhos realizados na construção do processo de validação. Na quarta seção, "Resultados", apresenta-se o que foi obtido até a conclusão deste artigo. Por fim apresentam-se comentários em "Conclusões", a Bibliografia utilizada em "Referências" e complementos na forma de "Apêndices".

2. Referencial Teórico

Esta seção apresenta um resumo da fundamentação teórica utilizada na realização deste trabalho. Cada subdivisão apresenta sobre os temas mais relevantes na busca pelo objetivo principal. Quanto a contribuições de trabalhos anteriores, observa-se que uma consulta realizada em bases de dados como a Biblioteca Digital Brasileira de Computação e a Biblioteca Digital Brasileira de Teses e Dissertações não retornou nenhum trabalho semelhante, sugerindo pioneirismo.

2.1. Processos

Como anteriormente afirmado, a implantação de processos no ambiente de desenvolvimento de produtos têm sido um dos fatores que contribuiu para o aumento da qualidade dos produtos produzidos tratando-se atualmente de uma peça fundamental a qualquer empresa da indústria ao menos desde os tempos da Revolução Industrial. A onipresença de processos é tal que se pode afirmar que não há qualquer produto ou serviço relevante nesse meio sem que haja um processo empresarial associado [Gonçalves, 2000]; todo trabalho importante realizado numa empresa moderna faz parte de algum processo [Graham e Le Baron, 1994]. Tal presença pode ser ilustrada por meio das figuras 1 e 2 onde, de três maneiras diferentes, ilustra-se a divisão de uma empresa com "processos" recebendo uma parte significativa da composição organizacional da mesma.

Figura 1. Organização de uma empresa

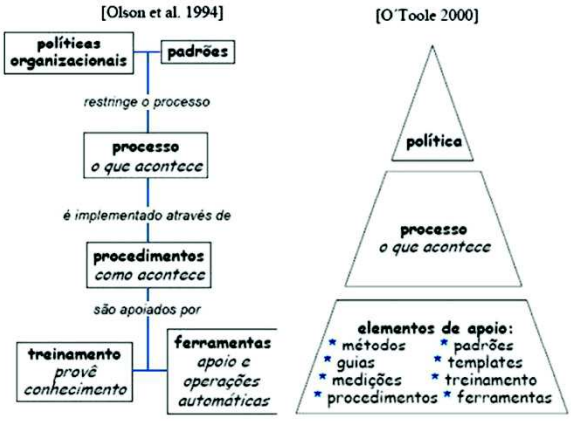

Figura 2. Organização de uma empresa [Becker, 2014]

Há várias maneiras como processos podem ser definidos. Segundo [Becker, 2014], um processo é um "conjunto de passos parcialmente ordenados, constituídos por atividades, métodos, práticas e transformações, usado para atingir um objetivo". Esta, porém, não é a única maneira que o termo tem sido definido no ambiente empresarial. Além de outras duas definições citadas pelo mesmo autor, tem-se aquela apresentada em [Salvino, 2014] onde, já contextualizado ao ambiente computacional, o autor define "processo de software" como "o que as pessoas fazem para um determinado propósito, utilizando suas habilidades e conhecimento, com o apoio de artefatos, ferramentas e outros recursos, para produzir software e seus produtos associados". Em suma, um processo é o que se tem quando se desenvolve um produto de uma maneira não aleatória, isto é, de uma maneira não baseada em improvisos.

Desenvolver produtos de uma maneira não aleatória, todavia, não significa que estejam sendo desenvolvidos de uma maneira realmente adequada e eficiente. É neste aspecto que se discrimina processos *imaturos* de processos *maduros*. Tais tipos se diferenciam pelas características apresentadas numa série de áreas tangidas por eles tais como definição de papéis e responsabilidades de uma equipe, reação a problemas, confiabilidade e previsibilidade. Em resumo, a existência de um processo imaturo provavelmente será superior em seus benefícios comparado ao desenvolvimento aleatório, mas não será capaz de extrair ao máximo os ganhos que podem ser obtidos quando processos são devidamente definidos e implantados no ambiente empresarial.

Aplicar processos a uma empresa consiste numa tarefa dividida em duas partes: a definição dos processos e a sua execução [Becker, 2014]. A primeira parte consiste em estabelecer, de forma precisa, a maneira como determinada tarefa será executada. Embora isso possa ser definido em viva voz por comum acordo entre as partes ou registrado de maneira simplória, é recomendável que um documento formal seja criado contendo as informações relevantes daquilo que foi definido, essa distinção inclusive sendo umas das grandes diferenças entre processos imaturos e maduros. Quanto à escrita deste documento, [Becker, 2014] lista os seguintes campos a serem incluídos:

- Nome;
- Objetivo;
- Responsável pelo processo;
- Atividades;

- Participantes;
- Responsabilidade por atividade;
- Critérios de entrada ou saída;
- Políticas e regras;
- Fluxo de atividades;
- Guias de uso (*guidelines*);
- Indicadores de desempenho.

Embora seja necessária como um passo inicial dentro da atividade de implantar processos em uma empresa, a definição de um processo não é uma tarefa estática a ser realizada uma única vez e nunca mais alterada, cabendo aos responsáveis pela sua criação a realização de atualizações na medida em que a experiência de uso proporciona novas ideias e sugestões de correções. Quando devidamente aplicado, os benefícios que se podem esperar da definição de um processo incluem [Becker, 2014]:

- Facilita o entendimento, treinamento e comunicação com os envolvidos;
- Permite o aperfeiçoamento do processo, através de implementação de melhorias;
- Permite a reutilização de definições e artefatos;
- Facilita a gestão do conhecimento e capital intelectual da empresa;
- Permite o controle e gerenciamento do processo;
- Auxilia na execução automatizada do processo.

Uma vez que uma equipe definiu um processo, é esperado que este seja executado. Em consonância com o que foi mencionado anteriormente, entre os benefícios esperados desta tarefa, encontram-se [Becker, 2014]:

- Padronização na execução das atividades e nos produtos gerados;
- Aumento da qualidade e confiabilidade de produtos e serviços;
- Redução na dependência de pessoas específicas para o sucesso dos projetos;
- Visão de fluxo de valor;
- Adoção disciplinada de novas tecnologias;
- Papéis e responsabilidades claramente definidos.

Dados os benefícios atribuídos à aplicação de processos dentro do ambiente empresarial e às exigências de um mercado competitivo e exigente, conclui-se que é desejável a todo grupo empreendedor que procure definir e seguir processos quando da realização de suas atividades. Naturalmente tais processos serão diferentes e exigirão esforços distintos, tanto quanto da sua definição quanto execução, dependendo da área a que se destinam, cabendo à empresa definir aquilo que lhe é mais propício segundo a sua realidade, sua visão e missão.

2.2. Testes e Validação

Entre todas as atividades comuns à busca por qualidade nos produtos comercializados no mercado, os testes encontram-se nos primeiros níveis numa escala de importância. Mesmo quando os produtos produzidos por uma empresa são tais que não permitem uma checagem minuciosa unitária, como no comércio de bens líquidos ou de produção em larga escala, testes são realizados em unidades representativas para auferir se o que está sendo produzido está de acordo com os padrões de qualidade estabelecidos pela empresa. Isso não é diferente na indústria de software, esta responsável por produzir um tipo de produto que, devido à sua alta complexidade tradicional, está naturalmente

propenso a conter uma grande quantidade de erros mesmo quando os processos de desenvolvimento estabelecidos são bons e estão sendo corretamente seguidos. Apesar de serem seguidamente negligenciados, os testes de software são parte essencial para qualquer grupo desenvolvedor que deseje entregar ao seu cliente um produto que atenda aos requisitos estabelecidos.

Embora o termo seja de uso comum, há várias maneiras como a palavra "teste" tem sido definida mesmo dentro do ambiente de software. Por exemplo, [Pimentel, 2014] define "teste de software" como "o processo de avaliação de um sistema ou componente sob condições específicas, cujos resultados são observados, registrados e analisados". Já [Delamaro *et. al.,* 2007] entende "teste" como a atividade dinâmica de execução de um programa ou modelo utilizando algumas entradas em particular e verificando se o seu comportamento está de acordo com o esperado. Levando-se tais definições em consideração além de outras como a em [ASTM, 2003], pode-se dizer que um teste, no contexto de software, é toda atividade que possui as seguintes condições necessárias e, juntas, suficientes:

- Ser ativa (ser exercida sobre o produto);
- Envolver o uso do produto ou parte dele que já tenha sido desenvolvido (nunca envolve a produção de novo código);
- Visa a coleta de resultados de uso para comparação com uma definição pré-estabelecida de resultados esperados (formalizada ou não);

A principal justificativa para a realização de testes jaz na inequívoca observação de que todos os seres humanos estão sujeitos a cometer erros que, no código, poderão se apresentar como instruções inadequadas (os *bugs*). Tais defeitos não se limitam a instruções que não realizam o que deveria ser feito, mas também àquelas que fazem o que não deveria ser feito ou o fazem de maneira inadequada. Se tais pedaços do código forem executados, uma falha irá acontecer, isto é, o software não irá se comportar como desejado podendo incorrer em sérios prejuízos para o seu usuário [Pimentel, 2014].

Todavia encontrar erros num programa não é a única razão porque testes podem ser realizados, como as condições acima indicam. Quando se testa o software, o que se obtém é mais conhecimento a seu respeito, isto podendo ser utilizado de várias maneiras: para vindoura correção e aprimoramento, para avaliá-lo com o fim de escolher comparativamente uma ferramenta para determinadas atividades ou para avaliar se o produto entregue ao cliente atende àquilo que foi solicitado em contrato [Pimentel, 2014].

Em função das diferentes motivações por traz da realização de testes bem como das diversas partes das quais os softwares são geralmente compostos, é comum que se divida a atividade de teste em várias categorias de acordo com parâmetros discriminativos que variam de acordo com os teóricos do ramo. Uma das divisões mais comuns gira em torno da utilização do conhecimento do código como parte da execução dos testes: se a estrutura de controle do projeto procedimental é utilizada, denomina-se teste "de caixa branca"; já se a execução ignora o código, o teste é dito "de caixa preta". Devido às suas distintas naturezas, o primeiro pode ser utilizado para garantir que a operação interna do produto tem um desempenho de acordo com as especificações e que os componentes internos foram adequadamente postos à prova enquanto que o segundo serve para demonstrar que cada função específica que um produto projetado deveria executar é totalmente operacional [Pressman, 1995].

Com base no tipo de teste a ser definido, os testadores podem utilizar-se de diferentes técnicas para a criação dos chamados "casos de teste". Estes consistem de um conjunto de informações prescritivas utilizadas para a realização de um teste, [Pimentel, 2014] isto é, o conjunto de entradas específicas que o testador irá tentar e os procedimentos que ele irá seguir quando testando um software [Patton, 2001]. Contendo um conjunto de valores de entrada, condições de execução, resultados esperados e pós-condições de execução, desenvolvidos para cobrir certas condições de avaliação [Pimentel, 2014] [IEEE, 1990], além de eventualmente outras informações de acordo com as preferências de cada empresa, eles são geralmente criados em grupos a serem executados dentro de uma "bateria" ou "rodada de teste" seguido alguma estratégia definida pela empresa.

Várias técnicas de criação de casos de teste para ambos os tipos foram elaboradas pela academia e pela indústria. Quanto aos de caixa preta, sobre os quais jaz o foco deste trabalho, destacam-se:

- **Particionamento de equivalência:** consiste na divisão do domínio de entrada de um programa em classes de dados a partir das quais os casos de teste podem ser derivados [Pressman, 1995]. Em outras palavras, ao invés de se construir um número grande de casos de teste cobrindo cada possível valor que uma determinada variável de um programa poderia receber, escolhe-se algumas variáveis representantes de grupos semânticos inteiros;
- **Análise de valor limite:** consiste na criação de casos de teste que analisam os valores fronteiriços nas situações em que uma determinada variável possui faixas de aceitação do domínio de entrada [Pressman, 1995]. Complementar ao particionamento de equivalência, fundamenta-se sobre o conhecimento de que um número maior de erros tende a ocorrer nas fronteiras dos domínios de entradas.

Ambas as técnicas mencionadas pressupõem uma abordagem de criação de teste orientada à especificação formal. Nesta maneira eles são criados tendo-se em vista os requisitos definidos no escopo do produto. Essa não é, todavia, a única maneira possível para se realizar tal tarefa; embora seja inimaginável de acontecer numa grande companhia, é possível que muitas das especificações de um produto ou suas partes sequer estejam apropriadamente definidas em documentação formal inviabilizando a criação de casos de teste por esse meio. Nesta situação somente possuindo um conhecimento aprofundado do produto será possível fazer a tarefa da criação dos casos de teste que naturalmente ficaria delegada aos desenvolvedores e não à equipe de testadores.

Caso os casos de teste sejam criados com base em requisitos formalmente estabelecidos, é provável que a sua organização numa ferramenta de gerenciamento de casos de teste seja relativa àqueles: grupos serão criados para cada especificação contendo todos os casos de teste que visam testar aquela parte do produto. Não possuindo especificações formais, há várias maneiras como os casos poderiam ser organizados. Uma maneira seria dividindo-os por seu relacionamento com os diversos subsistemas. Num produto como o Analisador RE8000, isso significaria dividi-los em casos de teste pertinente ao sistema GPS, à abertura de telas, ao sistema de registro de medições, etc.. Outra organização possível, embora muito mais genérica, é

considerando a natureza das partes do sistema: se estáticas, como figuras ou posicionamento dos itens, ou dinâmicas, como animações ou abertura de telas.

Uma última proposta de organização divide um produto de software em três partes segundo os critérios de funcionalidade e de interação com o usuário: um grupo é formado por todos os itens que não são funcionais (isto é, itens puramente visuais), um segundo grupo é formado pelos que são funcionais e que estão sujeitos à interação do usuário enquanto um terceiro é composto pelos funcionais que não estão sujeitos a tal interação. Elaborada para este trabalho, esta abordagem facilita a organização dos casos de teste bem como conduz o seu processo de criação ao propor tipos pré-determinados de casos de teste para cada item do produto que está sendo avaliado de acordo com a sua natureza.

Para os testes de itens puramente visuais, naturalmente descartáveis quando o produto não possui uma interface de usuário e difíceis de serem realizados de forma automatizada, os tipos de casos de teste propostos seriam os seguintes:

- **Implantação:** verificar se todos os itens gráficos estão presentes (e.g. se o ícone de um botão não está faltando);
- **Correspondência:** verificar se as imagens, os textos e as funcionalidades correspondentes estão semanticamente corretas (e.g. se um ícone que deveria representar o status de conexão com o GPS está mostrando uma imagem que representa essa conexão ao invés de, por exemplo, um símbolo de bateria);
- **Posicionamento:** se todos os itens presentes estão nos seus devidos lugares bem como se estão alinhados corretamente e com as devidas margens;
- **Limites:** se nenhum texto ou imagem está ficando para fora da área na qual deveria aparecer;
- **Coloração:** verificar a adequação das cores, como se há algum caso em que a cor de fundo de um texto é parecida com a cor do mesmo dificultando a sua leitura e se a cor utilizada para transmitir uma mensagem é adequada para o tipo de mensagem sendo dada (e.g. cor verde para uma mensagem positiva e vermelha para uma negativa);
- **Legibilidade:** se o tipo e o tamanho das fontes permitem adequadamente a leitura dos textos bem como se as imagens não estão pequenas demais ou indiscerníveis;
- **Língua:** se as mensagens textuais estão corretamente escritas e são compreensíveis e se as traduções são correspondentes aos textos originais;
- **Padrões:** se os itens estão de acordo com o padrão gráfico de todo o sistema (e.g. se as cores de certo tipo de botão são as mesmas ao longo de todas as telas).

Os itens funcionais sujeitos à interação do usuário cobrem todos os eventos de interação que o usuário possa ter com o produto através dos quais ele influencia na condução do mesmo. Em interfaces gráficas isso se traduz em itens como botões e campos de edição bem como as opções de menus em aplicativos tipo "terminal". Para estes itens, o método desenvolvido divide os tipos de casos de teste dependendo do tipo de item sendo testado:

- **Botões e assemelhados:** testes de "interação e reação" para todas as circunstâncias de uso possíveis (e.g. abertura de uma página ao se clicar sobre um *hyperlink*);
- **Movíveis:** verificar se todos os itens passíveis de *drag and drop* podem ser movidos nas condições adequadas;
- **Selecionáveis:** testes de "seleção e reação" para todas as circunstâncias de uso possíveis;
- **Campo de inserção numérica:** para tais campos deve se gerar casos de teste suficientes para cobrir as seguintes possíveis entradas, quando aplicáveis:
    - Valores limites se o campo aceita intervalos de aceitação (se o campo só aceita valores discretos, deve-se criar um caso de teste para cada valor possível);
    - Número zero e valor "nulo";
    - Combinações possíveis permitidas por máscaras de entrada;
    - Números reais (em campos para números inteiros exclusivamente);
- **Campo de inserção de texto:** novamente para tais campos deve se gerar casos de teste suficientes para cobrir as seguintes possíveis entradas, quando aplicáveis:
    - Tamanho de texto máximo e mínimo;
    - Uso de caracteres especiais:
        - Apenas caracteres especiais;
        - Caracteres especiais com letras;
        - Caracteres isolados;
        - Caracteres especiais específicos (quando alguns são admissíveis e outros, não);
        - Espaços;
    - Presença de números;
    - Combinações possíveis permitidas por máscaras de entrada;
    - Campo vazio
- Acessibilidade: se uma janela impede ou não o acesso a outras janelas;

Por fim tem-se o grupo dos itens funcionais que não estão sujeitos à interação do usuário, i.e., aqueles que funcionam de forma independente dos comandos do usuário ainda que sob efeito de configurações estabelecidas por ele. Um típico exemplo seria um relógio que deve se atualizar à medida em que o tempo passa. Como tais itens podem se apresentar nos mais variados tipos e geralmente são únicos dentro do produto, é difícil de se conceber tipos genéricos de casos de teste para eles. Dessa forma nenhum tipo de caso de teste é prescrito ficando a cargo da equipe responsável criá-los segundo técnicas tradicionais.

Além da divisão de testes entre "de caixa branca" e "de caixa preta" feita com base no uso ou não do código de um programa, autores como [Sommerville, 2011] e [Pressman, 1995] têm discriminado testes com base em critérios como quem os realiza e qual o seu objeto de estudo. As novas categorias que surgem com o uso destes critérios, como os "testes de aceitação", "unitários" e "de sistema", são abordadas por Pressman dentro do contexto de "estratégia de teste de software", esta definida como "a integração de técnicas de projetos de casos de teste numa série bem definida de passos que resultam na construção bem-sucedida de um software" [Pressman, 1995]. A ideia é que a equipe responsável pelos testes de um produto não crie simplesmente vários casos de

teste utilizando-se das técnicas anteriormente mencionadas, mas, considerando o produto e suas partes bem como um objetivo ao qual uma determinada rodada de testes se destina, defina um conjunto de casos a serem executado em determinada ordem e por determinados testadores de maneira organizada segundo os objetivos que se tem em mente.

No que diz respeito a tais objetivos, costuma-se discriminar dois objetivos distintos trabalhados dentro da área mais ampla da busca pela qualidade de um produto de software na qual a atividade de teste se inclui, a da "Verificação e Validação" (V&V) [Pressman, 1995]. Enquanto que a primeira pode ser entendida como "o conjunto de atividades que garantem que o software implemente corretamente uma função específica", a segunda envolve atividades que buscam garantir que o produto desenvolvido foi concluído segundo as exigências do cliente [Pressman, 1995] [Sommerville, 2011], isto é, que o produto desenvolvido irá cumprir o seu uso pretendido quando colocado no seu ambiente desejado [CMMI, 2010]. Nas célebres palavras de Boehm, a Validação é atividade que procura determinar se o que foi desenvolvido "é o produto certo" [Boehm, 1981].

Dados os seus conceitos, tanto a Verificação quanto a Validação naturalmente englobam uma série de atividades que vão além da realização de testes tais como revisões técnicas formais, monitoração do desempenho, análise de algoritmos, entre outras [Pressman, 1995]. No caso da realização de testes aplicados à Validação, as suas atividades também podem ser realizadas a todos os aspectos envolvendo o produto tais como operação, manutenção e serviço de suporte [CMMI, 2010]. Uma execução plena de ambas as áreas, portanto, envolveria a realização de todas estas tarefas o que está fora do escopo deste trabalho. Antes, traçou-se como suficiente dentro dos objetivos da Embrasul Indústria Eletrônica Ltda. em torno do Analisador RE8000 que se realizasse apenas a validação do aparelho por meio de testes do tipo "caixa preta" segundo proposto pelo modelo de maturidade CMMI-Dev 1.3 abordado na próxima seção.

2.3. Processo de Validação

A validação de um produto de software não constitui numa atividade rígida podendo ser planejada e executada tanto de forma desorganizada quanto como um processo claramente definido. Quanto ao caso da implantação de um processo de validação, já há alguns modelos disponibilizados e usados pelo mercado. Exemplos de tais modelos incluem o presente no Melhoria de Processos do Software Brasileiro (MPS.BR), um modelo de qualidade de processo desenvolvido por algumas entidades nacionais e gerenciado pela Softex [Softex, 2012] visando a melhoria da qualidade e competitividade dos produtos de software nacionais no mercado mundial [Silveira, 2011], e o do modelo de maturidade CMMi-Dev versão 1.3, desenvolvido pelo *CMMI Institute* e escolhido pela Embrasul Indústria Eletrônica Ltda. Como referência para a criação do processo de validação do RE8000.

O processo segundo este modelo é organizado em partes que vão se ramificando até chegar às atividades do processo de validação propriamente ditas. Tais atividades são executadas sobre Produtos de Trabalho e produzem Resultados que deverão ser utilizados nas fases subsequentes da validação ou mesmo após esta. Detalhar cada uma destas atividades está fora do escopo deste trabalho cabendo aqui apenas uma breve menção ao que é feito em cada uma de suas etapas.

O processo proposto é dividido em duas partes chamadas "Objetivos Específicos" (OE) [CMMI, 2010]: "Preparação para a Validação" e "Validação de Produto ou Componentes de Produto". A primeira OE contém todas as atividades anteriores ao início da execução do processo sendo dividida em três partes denominadas "Práticas Específicas" (PE) [CMMI, 2010]: "Selecionar Produtos para Validação", "Estabelecer o Ambiente de Validação" e "Estabelecer Procedimentos e Critérios de Validação". As subpráticas de seleção de produto são [CMMI, 2010]:

1. Identificar os princípios principais, funcionalidades e fases para a validação do produto ou componente de produto ao longo da vida do projeto.
2. Determinar que categorias das necessidades do usuário final serão validadas (comportamento, manutenção, treinamento, etc.).
3. Selecionar o produto e componentes de produto a serem validados.
4. Selecionar métodos de avaliação de validação de produto ou componente de produto.
5. Revisar as seleções, limitações e métodos com os *stakeholders* relevantes.

Já as subpráticas da segunda PE, "Estabelecer o Ambiente de Validação", são as seguintes [CMMI, 2010]:

1. Identificar os requerimentos para o ambiente de validação.
2. Identificar produtos fornecidos pelo cliente.
3. Identificar equipamentos e ferramentas de teste.
4. Identificar recursos de validação que estão disponíveis para reuso e modificação.
5. Planejar a disponibilidade dos recursos em detalhe.

Por fim, a terceira Prática Específica é compreende as seguintes subpráticas [CMMI, 2010]:

1. Revisar os requisitos do produto para garantir que problemas afetando a validação do produto ou componente de produto são identificados e resolvidos.
2. Documentar o ambiente, cenário de operação, procedimentos, entradas, saídas e critérios de validação do produto ou componentes de produto selecionados.
3. Avaliar o design na medida em que ele amadurece no contexto do ambiente de validação para identificar problemas de validação.

A segunda OE compreende todas as atividades de execução da validação e é dividida em duas Práticas Específicas [CMMI, 2010]: "Realizar a Validação" e "Analisar os Resultados da Validação". Quanto a primeira PE, nenhuma subprática é especificada uma vez que ela é constituída exclusivamente da execução dos testes definidos nas etapas anteriores. Já quanto a "Analisar os Resultados da Validação", as atividades prescritas são as seguintes [CMMI, 2010]:

1. Comparar resultados reais com esperados.
2. Baseado nos critérios de validação estabelecidos, identificar produtos e componentes de produtos que não são adequadamente executados em seus ambientes operacionais pretendidos ou identificar problemas com métodos, critérios ou ambiente.
3. Analisar dados da validação buscando por defeitos.
4. Registrar os resultados da análise e identificar problemas.
5. Usar os resultados da validação para comparar medições e desempenho reais para o uso pretendido ou necessidade operacional.

6. Providenciar informação sobre como defeitos podem ser resolvidos (incluindo quanto a métodos, critérios e ambiente de validação) e iniciar ação corretiva.

2.4. Ferramentas de Suporte

Paralelo ao crescimento da Engenharia de Software surgiu uma série de ferramentas destinadas a providenciar suporte à realização dos diversos passos da construção de um software. A área de testes e especialmente a parte referente a processos de validação não ficou à margem deste fenômeno: muitos aplicativos, tanto pagos quanto gratuitos, foram desenvolvidos especificamente para esta área. As dezenas [Kayani, 2015] de ferramentas construídas diferenciam-se quanto às características disponibilizadas que permitem não apenas um controle avançado dos casos de testes, como auxílio na concepção dos mesmos e na realização das rodadas de testes. Exemplos populares destas ferramentas incluem: o Testopia [MDN, 2015], uma extensão do Bugzilla e também desenvolvida pela Mozilla; o TestLink [TestLink, 2015], uma plataforma web gratuita e o Tarantula [Tarantula, 2015], também gratuito.

Embora uma menção exaustiva às ferramentas disponíveis e às suas possíveis características esteja fora do escopo deste trabalho, convém apresentar algumas das *características* que são passíveis de consideração quando da escolha de qual ferramenta utilizar. Ter o conhecimento destas diferenças é o primeiro passo para se poder fazer a escolha do que se pretende ter numa ferramenta de suporte a testes, este sendo o passo inicial necessário antes da escolha final [Inflectra, 2014]. A seguinte lista apresenta alguma destas características com breves comentários a seu respeito [InformUp, 2015]:

- **Sistemas operacionais compatíveis:** a possibilidade de executar o software no sistema operacional de preferência;
- **Natureza da licença:** pode não ser aceitável a uma companhia investir uma grande soma de dinheiro numa ferramenta comercial se houver equivalentes gratuitos aceitáveis;
- **Natureza da ferramenta:** diz respeito a ser um aplicativo web, possível de ser acessado por qualquer pessoa que possua a devida autorização por um link em algum *browser*, ou se é um aplicativo para *smartphone* ou computador *desktop* que requer instalação e só pode ser acessado por máquinas específicas;
- **Suporte e atualizações:** aplicativos antigos que não são mais atualizados por seus desenvolvedores ou a falta de suporte técnico destes são riscos para uma equipe de desenvolvimento especialmente quando se trata de ferramentas complexas, com mais propensão à existência de defeitos, e que não são *open source* inviabilizando a correção de *bugs* pelos seus usuários. A falta de suporte adequado na forma de manuais e guias de uso também constitui um empecilho ao pleno aproveitamento de um aplicativo;
- **Conexão com outras ferramentas:** a empresa pode fazer uso de outras ferramentas em seu processo produtivo, situação na qual é vantajoso ter facilidade para troca de dados entre elas. A possibilidade de transferir dados com facilidade para uma planilha ou documento ou reportar defeitos a um programa para esse fim, como o Bugzilla, pode significar horas de trabalho ganhas;
- **Facilidade no uso:** em suma, quanto mais amigável é a interface de um sistema e mais intuitivo o uso, melhor. Uma ferramenta com menos funcionalidades, mas cujo uso seja mais agradável e fácil pode ser preferível aos seus usuários ao oposto

- **Dependências externas:** o software pode requerer a instalação de algum aplicativo a mais que seja indesejável pela empresa;
- **Possibilidade de customização:** algumas ferramentas já vêm com um sistema mais inflexível obrigando aos seus usuários utilizarem-na da maneira como foi projetada, enquanto outras permitem certo nível de customização

Embora o item do Manifesto Ágil que afirma serem indivíduos e interação entre eles mais importantes do que processos e ferramentas [Sommerville, 2011] [Agile, 2001] ser entendido por muitos como correto, a escolha de uma ferramenta inadequada pode se tornar um problema completamente evitável para o projeto. Para que isso seja evitado, é importante que a equipe responsável pela escolha das ferramentas não apenas tenha a opinião dos seus futuros usuários, como o cuidado de ter em mente o que se enquadra mais à realidade de cada projeto.

3. Metodologia

Nesta seção são descritos os procedimentos realizados no desenvolvimento deste trabalho.

3.1. Seleção da Ferramenta

Tendo em vista a teoria sobre ferramentas de suporte ao gerenciamento de casos de teste apresentada na seção "Referencial Teórico", foi realizada uma pesquisa na Internet visando encontrar o maior número possível de ferramentas candidatas a gerenciar os casos de teste do processo de validação do software da interface do Analisador RE8000.

Essa pesquisa se deu em três etapas cada qual empregando um grupo de critérios seletivos para a obtenção dos seus resultados: a primeira consistiu na criação de uma lista compreensiva das ferramentas existentes; a segunda serviu para criar um subgrupo de ferramentas a partir da lista criada no passo anterior; por fim a terceira parte compreendeu uma avaliação final, mais subjetiva, quando a ferramenta vencedora foi escolhida.

Na primeira etapa o único critério considerado foi que a ferramenta deveria ser gratuita. Esse critério foi estabelecido pela equipe da empresa como mandatório por não haver a possibilidade de deslocar recursos adicionais para este trabalho.

Na segunda etapa os critérios levados em consideração foram:

- Disponibilidade para instalação em Linux, o sistema utilizado pelos membros da equipe de Pesquisa e Desenvolvimento da Embrasul Indústria Eletrônica Ltda.;
- Dependência de softwares secundários não utilizados pela equipe;
- Acessibilidade ao uso (algumas ferramentas poderiam não ser possíveis de sequer serem obtidas);
- Status de desenvolvimento (se já tinham sido descontinuadas ou continuavam recebendo manutenção);

Por fim a terceira etapa, que envolvia uma tentativa de instalação da ferramenta, considerou os seguintes critérios:

- Sucesso na instalação;
- Interface intuitiva e facilidade no uso;

- Ausência de qualidades menores indesejáveis (como limitação de número de casos de testes possíveis de serem criados);

Todo o processo avaliativo (especialmente a última etapa) foi realizado tendo como *benchmark* o *Google* Tabelas. Planilhas eletrônicas, embora simples, provém muito do que se deseja de uma ferramenta de armazenamento de casos de teste. Além de muitas serem gratuitas e passíveis de visualização em qualquer computador via Internet independente do sistema operacional, como é o caso das planilhas do *Google*, elas são muito flexíveis permitindo não apenas a criação de tabelas customizadas como também adição de gráficos e outros utensílios mais avançados. Além disso, parte da equipe da Embrasul Indústria Eletrônica Ltda. envolvida na utilização do processo de validação sendo construído já tinha experiência nesta ferramenta para este mesmo fim, o mesmo não sendo verdadeiro para quase nenhuma das demais ferramentas.

Por estas razões concluiu-se que o *Google* Tabelas, embora tenha alguns problemas como a necessidade de se montar as planilhas de casos de testes desde o começo, poderia ser tranquilamente utilizado para o fim pretendido. A pesquisa por outras ferramentas foi, portanto, justificada como o meio para responder à questão sobre se haveria alguma outra ferramenta que seria significativamente superior ao ponto de justificar a sua utilização em detrimento de uma planilha no sistema do *Google*.

3.2. Implantação do Processo

O processo de validação criado para validar o Analisador RE8000 foi baseado no exemplo de processo presente no modelo de maturidade CMMi-Dev 1.3 como resumido no referencial teórico contendo uma série de modificações em relação àquele. Tais diferenças foram justificadas com base na necessidade de se adaptar o modelo à realidade tanto do ambiente de desenvolvimento do RE8000 quanto do próprio projeto.

Tendo sido disposto numa série de passos cronológicos como no processo de validação do CMMi-Dev, as modificações incluíram a retirada de algumas das "fases" de trabalho propostas pelo modelo, mudanças de lugar de outras e também adição de novas fases. Um exemplo de tais modificações é aquela presente na etapa "Definir entidades da validação", equiparável ao Processo Específico 1.1, que inclui as entidades chamadas "passivas" (as que sofrem ação dos agentes da validação) não sendo esta parte do Processo Específico 1.1 original do CMMi-Dev.

Quanto à criação dos casos de teste, a parte do trabalho que mais recebeu ênfase por parte da empresa e que consequentemente concentrou os maiores esforços, optou-se por usar uma planilha do Google Tabela como ferramenta para o seu gerenciamento até que um software melhor do que os avaliados fosse desenvolvido. Essa decisão é comentada em maiores detalhes na seção "Resultados". Nesta tabela foram criados três abas, uma para cada grupo de testes cada qual relativo a uma classe de caso de teste como comentado no "Referencial Teórico". A decisão por organizar dessa maneira se baseou tanto no fato que tal organização permite uma cobertura completa das funcionalidades do software da interface do usuário (e, por conseguinte, de todo o Analisador) como também porque o desenvolvimento deste se deu em grande parte sem o registro formal dos seus requisitos, praticamente inviabilizando a criação dos casos com base em requisitos formais. Por fim, também foi adicionada à tabela uma aba inicial para mostrar as estatísticas de cada rodada de testes.

Com a finalização da criação da tabela e a escrita dos casos de teste, o processo final foi concluído. Este com todos os seus passos bem como uma parte da planilha de casos de teste são mostrados na seção "Resultados".

Após o processo ter sido definido, prosseguiu-se com a sua implantação e teste prático a ser executado com uma rodada de testes. Devido a limitações de cronograma, não foi possível executar o processo por inteiro; os únicos itens que foram executados na única rodada de validação realizada foram os relativos à criação dos casos de teste, os de preparação para a realização da validação e os referentes à execução do próprio. Esta, por sua vez, não contou com a realização de todos os mais de 550 casos criados, mas tão somente os 273 que foram classificados como mais importantes.

3.3. Avaliação do Processo

Devido à natureza do trabalho, não havia como testar se o processo de validação havia sido bem construído que não uma simples revisão semelhante a um *walkthrough*. A única opção concebida para se obter algum *feedback* foi uma consulta aos testadores quanto às suas opiniões sobre o processo e em especial sobre os casos de teste por meio de um questionário. Este, passado a eles na forma impressa, continha três perguntas, estas devendo ser respondidas com uma nota entre 1 e 5 com "1" sendo a nota mais baixa e "5" a nota mais alta, além de um campo para comentários livres. Tais perguntas foram elaboradas visando obter a percepção dos testadores quanto à completitude e organização do que fora desenvolvido.

O questionário encontra-se disponível como Apêndice A deste trabalho. Os resultados da avaliação realizada são apresentados na próxima parte deste artigo.

4. Resultados

Esta seção contém os resultados obtidos em cada parte dos trabalhados descritos na seção "Metodologia".

4.1. Seleção da Ferramenta

A primeira parte da pesquisa por ferramentas de suporte à validação resultou numa lista com vinte e nove (29) ferramentas em software ou *web* e dois (2) *templates* de planilhas do *Google* Tabelas. Tais ferramentas foram encontradas seja diretamente, pelo mecanismo de busca da *Google*, ou consultando-se listas disponíveis na *web* sobre o tema [Kayani, 2015] e todas satisfazem o critério de gratuidade estabelecido para esta fase da pesquisa.

Na segunda parte foram selecionados quatorze (14) das ferramentas da primeira lista mais um dos *templates* de planilhas do *Google* Tabelas. Neste estágio a maior parte das desclassificações se dera pela ferramenta não ser executável no sistema operacional Linux Ubuntu ou porque tinha sido descontinuada.

Por fim, na terceira parte, cujo objetivo era a seleção da ferramenta final, concluiu-se que nenhuma das ferramentas disponíveis era suficientemente superior a uma planilha customizada do *Google* Tabelas, o *benchmark* usado durante todo o processo avaliativo.

Todas as ferramentas que participaram deste processo avaliativo são mostradas na tabela 1 que também apresenta, ao lado de cada item, um breve comentário quanto à razão porque cada ferramenta foi rejeitada.

Quadro 1. Ferramentas analisadas com comentários

Ferramenta	Resultado da Avaliação
1. Fitnesse	Não funciona no Linux Ubuntu; Requerimentos indesejados
2. Incremental Scenario Testing	Fora do escopo
3. Litmus	Descontinuado/Sem suporte
4. Microsoft Test Manager	Não funciona no Linux Ubuntu; Requerimentos indesejados
5. MozTrap	Problemas na instalação
6. QAManager	Fora do escopo
7. QaTraq	Problemas na instalação
8. Radi	Problemas na instalação
9. RIACase	Não funciona no Linux Ubuntu
10. RTH	Descontinuado/Sem suporte
11. RTH-Turbo	Descontinuado/Sem suporte
12. RTMR	Problemas na instalação
13. Salome-TMF	Descontinuado/Sem suporte; Experiência do usuário ruim
14. Speed Test	Não funciona no Linux Ubuntu; Descontinuado/Sem suporte
15. Squash TM	Problemas na instalação
16. Tarantula Agile Test Management Tool	Problemas na instalação
17. Test Case Web	Descontinuado/Sem suporte; Problemas na instalação
18. Tesly	Descontinuado/Sem suporte
19. Test Analytics	Fora do escopo
20. Test Environment Toolkit	Fora do escopo; Experiência do usuário ruim
21. Test Link	Experiência do usuário ruim
22. TestAutomation	Não funciona no Linux Ubuntu
23. TestCube	Não funciona no Linux Ubuntu
24. Testitool	Descontinuado/Sem suporte
25. Testmaster	Problemas na instalação
26. Testopia	Problemas na instalação
27. Vienna 2/QABook	Não funciona no Linux Ubuntu
28. Webtst	Fora do escopo; Problemas na instalação
29. Xqual Studio	Requerimentos indesejados

30. Planilha Google Tabelas 1	Equivalente à criação de uma tabela customizada
31. Planilha Google Tabelas 2	Equivalente à criação de uma tabela customizada

Com base no resultado da pesquisa por uma ferramenta de gerenciamento de casos de teste, apresentado na seção de conclusões, a decisão tomada foi a de usar temporariamente uma planilha eletrônica enquanto, paralelamente, um software que preenchesse todos os requisitos usados na pesquisa seria desenvolvido.

O início da construção desse software se deu logo após o fim da pesquisa pela ferramenta, embora não tenha sido concluído até a data final da escrita deste artigo. Aquele, intitulado mTestManager (Figura 3), está sendo desenvolvido em Qt o que permitirá que seja compilado e usado em todos os sistemas operacionais mais comuns, notoriamente o Linux Ubuntu. Além dos *pop-ups*, a sua interface contém apenas uma tela dividida em duas partes, buscando-se com isso atingir o objetivo de se ter um aplicativo de fácil uso e de *design* limpo trazendo uma boa experiência para o usuário.

A possibilidade de adicionar milhares de subcategorias na árvore de casos de teste permite uma grande flexibilidade na organização dos mesmos, mas sua primeira versão não permitirá customização quanto aos campos de informação dos casos de teste. As informações inicialmente serão salvas num banco de dados, escrito em SQLite, armazenado localmente ou remotamente bastando haver conexão com o repositório.

O sistema também possibilitará a criação de rodadas de teste identificadas por nome único, testador entre outras informações que ficarão salvas no mesmo banco de dados onde os casos de testes se encontram. Dessa forma qualquer usuário do programa poderá carregar não só os testes de validação, como também uma rodada de validação com os resultados finais. Estes, por sua vez, poderão ser publicados num documento no formato PDF contendo dados estatísticos e gráficos com os resultados atuais de cada rodada de testes.

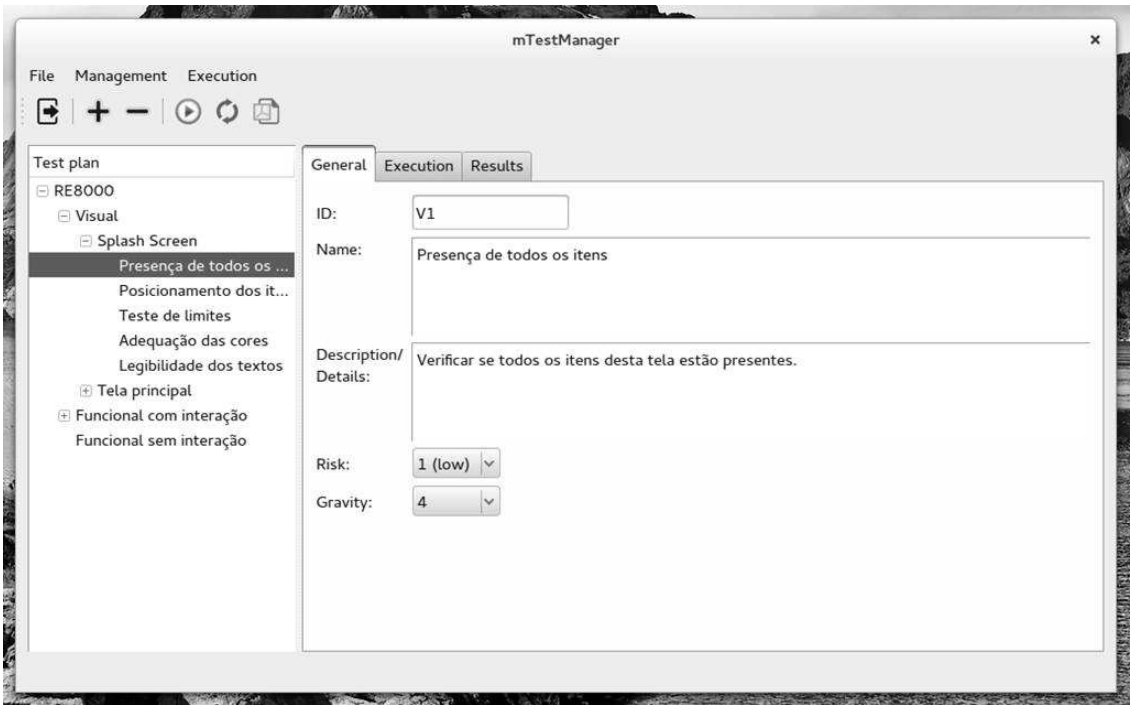

Figura 3. Interface do mTestManager

4.2. Implantação do Processo

Os passos que compõe o processo de validação para o Analisador RE8000 são os seguintes:

(OS 1) Parte 1: Preparar a validação

1. Preparação básica
    a. Definir documento em que plano de validação será redigido
    b. Definir espaço onde os documentos relevantes à validação ficarão armazenados
    c. Definir plano de reuniões de acompanhamento
    d. Avaliar orçamento
    e. Definir estratégia de validação
    f. Definir cronograma da validação
    g. Definir plataforma de gerenciamento de casos de teste
    h. Criar template de incidente para erros encontrados
2. (PE 1.1) Definir entidades da validação
    a. Selecionar itens à serem validados
        i. Entidades ativas
            1. Testadores
            2. Desenvolvedores provedor de suporte
            3. Softwares de automação de testes
        ii. Entidades passivas
            1. Produto ou partes de um produto a serem validadas
            2. Aplicativos e aparelhos como simuladores que formam ou emulam o sistema no qual o produto estará integrado
    b. Coletar os requerimentos e limitações para a realização da validação.
    c. Revisar as seleções, limitações e métodos com os *stakeholders* relevantes.
3. (PE 1.2) Estabelecer o ambiente de validação
    a. Identificar os requerimentos para o ambiente de validação
    b. Identificar recursos de validação que estão disponíveis para reuso e modificação
4. (PE 1.3) Definir procedimentos e critérios de validação
    a. Criar casos de teste ou revisar os já existentes
        i. Grupos de testes das seguintes categorias:
            1. Puramente visual
            2. Funcional com interação do usuário
            3. Funcional sem a interação do usuário
            4. Anexos
        ii. Itens dos casos de teste
            1. Fixos: Subgrupos, ID, Nome/Descrição, Risco/Gravidade, Dependências internas, Ambiente/Entrada, Roteiro, Saída (esperada),
            2. Editáveis pós execução dos testes: Resultado, Comentários, Executor
    b. Definir critérios de saída

(OS 2) Parte 2: Validar o produto

1. Preparar o ambiente e as entidades para a validação
    a. Preparar o local de validação (acomodações)
    b. Preparar entidades ativas
    c. Preparar entidades passivas
2. (PE 2.1) Executar validação
    a. Realizar as atividades de validação. Para cada caso de teste:
        i. Preparar execução do teste (montar o setup)
        ii. Executar parâmetros de entrada
        iii. Registrar resultados
        iv. Concluir com informações finais (e.g. passou, responsável, etc.)
    b. Registrar os desvios ocorridos durante a execução como apropriado: produzir relatório de incidente sempre que um erro é descoberto
        i. Itens do relatório: ID do teste falho (se aplicável), data da descoberta, rodada de validação no qual foi descoberto, resultados esperados, resultados obtidos (descrição do problema), severidade, prioridade de correção, status atual do incidente (aberto, aceito, etc.), comentários
3. (PE 2.2) Analisar resultados da validação
    a. Analisar dados da validação buscando por defeitos
    b. Registrar os resultados da análise e identificar problemas
    c. Produzir relatório contendo:
        i. Dados estatísticos da rodada de testes
        ii. Comentários sobre o próprio processo
    d. Adicionar conhecimentos aprendidos ao banco de dados da instituição alterando o processo de acordo
    e. Determinar próximos passos em reação aos resultados da validação
        i. Providenciar informação sobre como defeitos podem ser resolvidos e iniciar ação corretiva

A figura 4, por sua vez, mostra um *print screen* da planilha de casos de teste:

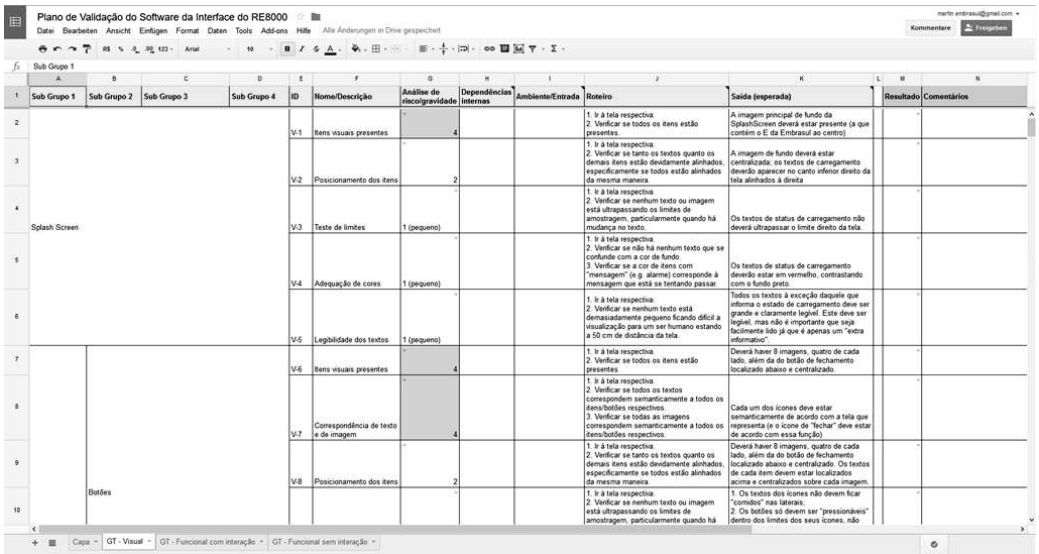

Figura 4. A planilha de casos de teste

4.3. Avaliação do Processo

A avaliação realizada por meio de um questionário resultou na aprovação plena do processo criado com destaque para o sistema de classificação dos casos de teste por nível de importância. A única sugestão dada pelo testador responsável pela execução dos testes foi quanto ao número de casos de testes de ações repetitivas no grupo de testes voltados a itens puramente gráficos. Segundo a sua observação, a disposição de vários casos de teste com os mesmos procedimentos podia ocasionar em erro ao dirigir o testador na suposição de que os próximos casos a serem realizados seriam "exatamente iguais aos anteriores" quando, em momentos específicos, alguma observação poderia estar presente. Por causa disso, foi sugerido que todos os testes gráficos fossem agrupados em casos de teste únicos.

5. Conclusões

O presente artigo apresentou o trabalho de implantação de um processo de validação para o Analisador de Energia RE8000, realizado nas dependências da empresa Embrasul Indústria Eletrônica Ltda.. Ao final, pôde-se observar que o objetivo principal de desenvolver um processo de validação para o Analisador RE8000 foi atingido embora restrições de criação, implantação e detalhamento tenham se apresentado fruto das limitações de cronograma e da realidade do setor de pesquisa e desenvolvimento da empresa. Por outro lado, não é possível auferir se os objetivos secundários também foram atingidos uma vez que isso dependerá da condução da empresa quanto às tarefas a serem realizadas após a conclusão deste trabalho podendo, inclusive, ser isso objeto de estudo no futuro.

Quanto à área da Qualidade de Software e mais especificamente o tema de Verificação e Validação, pode-se concluir que esta pesquisa acabou por trazer contribuições ao ramo especialmente na parte condizente à maneira de se conceber casos de teste. Embora não tenha sido o desejado e possa ter carecido de alguns itens a mais, o método de concepção de casos de teste proposto e usado neste trabalho permite uma organização mais avançada dos casos de testes sem a necessidade de haver documentos com as especificações formais do produto. Isso permite às empresas de menor porte e também as que trabalham com Métodos Ágeis realizarem o gerenciamento desta parte dos seus planos de validação sem precisarem enfrentar maiores problemas com a sua comum e menor ênfase em documentação.

Por fim, a realização deste trabalho acabou resultando no início de um novo projeto, o mTestManager, que uma vez concluído e disponibilizado à público poderá contribuir significativamente com o mercado de desenvolvimento de softwares ao possibilitar uma ferramenta simples e eficaz em cumprir a tarefa ao qual se destina: gerenciar casos e rodadas de teste de validação num sistema multiplataforma com dados acessíveis em várias máquinas por meio de uma interface amigável ao usuário.

Referências

Agile (2001) “Manifesto for Agile Software Development”, http://www.agilemanifesto.org/

ASTM (2003) “ASTM E 1301 Standard Guide for Proficiency Testing by Interlaboratory Comparisons”, http://www.astm.org/Standards/E1301.htm

Bartié, A. (2002) “A Garantia da Qualidade de Software”, Editora Campus, Rio de Janeiro

Becker, C. A. (2014) “Definição de Processos de Software”

Beizer, B. (1990) “Software Testing Techniques”, Van Nostrand Reinhold, 2º edição

Boehm, B. W. (1981) “Software Engineering Economics”, Prentice-Hall. ISBN 0-13-822122-7

Brennecke, A., Keil-Slawik, R. (1996) “History of Software Engineering”, https://www.dagstuhl.de/Reports/96/9635.pdf

CMMI (2010) “CMMI for Development, Version 1.3”, Software Engineering Institute, Carnegie Mellon University

Delamaro, M. E., Maldonado, J. C. e Jino, M. (2007) “Introdução ao Teste de Software”, Rio de Janeiro, Campus

Galagarra, O. (2014) “Melhoria do Processo de Software com Modelos de Maturidade CMMI e MPS.BR”

Garvin, D. (2002) “Gerenciando a qualidade: a visão estratégica e competitiva”, Editora Qualitymark

Gonçalves, J. E. L. (2000) “As empresas são grandes coleções de processos”, RAE Revista de Administração de Empresas, São Paulo, Jan/Mar v.40. n.1

Graham, M., Lebaron, M. (1994) “The horizontal revolution”, San Francisco, Jossey-Bass

IEEE, STD. 610 (1990) “IEEE Std. 610.12 IEEE Standard Glossary of Software Engineering Terminology”, The Institute of Electrical and Electronics Engineers, New York

Inflectra (2014) “How to Choose a Test Management Tool”, https://www.inflectra.com/Ideas/Whitepaper/How-to-choose-a-Test-Management-Tool.aspx

InformUp (2015) “5 Things to Consideration While Choosing a Test Case Management Tool”, http://www.informup.com/articles.aspx?5-Things-to-Consideration-While-Choosing-a-Test-Case-Management-Tool&article=3

Kayani, N. A. "Test management tools (29 found)", http://www.opensourcetesting.org/about.php, acesso em 31 ago. 2015

MDN (2015) “Testopia”, https://developer.mozilla.org/en-US/docs/Mozilla/Bugzilla/Testopia

Molinari, L. (2008) “Testes de Software”, São Paulo, Érica, 4º edição

Myers, G. (1979) “The Art of Software Testing”, New York, Willey

Naik, K. e Tripathy, P. (2008) “Software Testing and Quality Assurance”, John Wiley & Sons

Niquette, P. (1995) “Softword: Provenance for the Word ‘Software’”, http://www.niquette.com/books/softword/tocsoft.html, In: Sophisticated: The Magazine. ISBN 1-58922-233-4

Patton, R. (2001) “Software Testing”, Indianapolis, Sams. ISBN 0-672-31983-7

Pimentel, S. (2014) “Verificação e Validação”

PMBOK (2008) “Guia PMBOK - Um Guia do Conhecimento em Gerenciamento de Projetos”, Project Management Institute, Inc., 4º edição

Pressman, R. S. (1995) “Engenharia de Software”, São Paulo, Pearson Makron Books

Pressman, R. S. (2001) “Software Engineering: a practitioner’s approach”, Boston, McGraw-Hill, 5º edição

Salviano, C. F. (2014) “Modelos de Referência para Processos de Software”

Silveira, A. R. (2011) “O que é o MPS.br?”, https://www.oficinadanet.com.br/artigo/desenvolvimento/melhoria-de-processos-do-software-brasileiro--mpsbrSoftex (2010) “MPS.BR - Melhoria de Processo do Software Brasileiro - Guia Geral MPS de Software”, http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012-c-ISBN-1.pdf, ISBN 978-85-99334-48-5

Sommerville, I. (1982) “Software Engineering”, Addison-Wesley. ISBN 0-201-14229-5

Sommerville, I. (2011) “Engenharia de Software”, São Paulo, Pearson Prentice Hall, 9º edição

Tarantula, http://www.tarantula.fi/, acesso em 31 ago. 2015

TestLink, http://testlink.org/, acesso em 31 ago. 2015

Wirth, N. (2008) “A Brief History of Software Engineering”, http://people.inf.ethz.ch/wirth/Miscellaneous/IEEE-Annals.pdf

APÊNDICES

APÊNDICE A – Questionário de Opinião Sobre o Processo de Validação do RE8000

Questionário sobre o Processo de Validação do Analisador RE8000

Instruções

O presente questionário visa coletar a sua opinião sobre o Processo de Validação criado para o Analisador RE8000. As respostas às seguintes perguntas deverão ser dadas de acordo com o nível com que concordas ou discordas da pergunta num grau que vai de 1 a 5 com 1 sendo mais baixo e 5 sendo mais alto.

Perguntas

1. O que você achou dos campos dos casos de teste? (ID, Nome, etc.) Dê a sua nota expressando a sua aprovação.

1 2 3 4 5

2. Quão completos são os casos de teste? Se foi necessário criar novos casos, isso se deu porque alguma *feature* foi esquecida ou porque a organização não contemplava aquela *feature*? Dê a sua nota expressando o quão completo é o grupo de casos de teste criado.

1 2 3 4 5

3. De modo geral, entre "insatisfeito" e "satisfeito", qual seria a sua nota para o processo de validação do RE8000?

1 2 3 4 5

4. Escreva no espaço abaixo algum comentário sobre o processo de validação.

_____
_____
_____
_____
_____
_____