UMA PROPOSTA DE UM PROCESSO DE TESTE EM UMA EMPRESA DE PEQUENO PORTE

Juliano Barbosa Prettz, Cristiano Bertolini (Orientador)
Universidade Federal de Santa Maria – UFSM
Centro de Educação Superior do Norte - CESNORS
Frederico Westphalen – RS
prettzb@gmail.com

***Resumo*—Em geral, empresas de desenvolvimento de software de pequeno e médio porte utilizam meio informais para o controle de garantia e qualidade no desenvolvimento de software. Esta situação motiva empresas a introduzirem um processo de teste de software para elevar seu nível de qualidade em seus produtos. Teste de software é uma das técnicas de desenvolvimento que podem assegurar a qualidade dos produtos através da identificação de defeitos. Este trabalho apresenta a proposta de um processo de teste de software para uma empresa de pequeno porte. Como principal resultado, foi proposto um processo de teste baseado na metodologia ágil TDD onde foi incorporado outras técnicas de teste para adequar o processo proposto ao processo de desenvolvimento atual da empresa.**

Palavras-Chave: Teste de Software, TDD, Test Driven Development

***Abstract*—In general, small software companies use informal ways to quality control of their software development. This situation motivates companies to incorporate a software testing process to increase the quality of their products. Software testing is a development technique that can assure quality through the defects identification. This work presents a proposal of a software testing process for a small company. As main result, it was proposed a testing process based on the agile methodology TDD where it was included other testing techniques suited to the proposed process with the software development process used by the company.**

Keywords: Software Testing, TDD, Test Driven Development

I. INTRODUÇÃO

A indústria da tecnologia de informação está cada vez mais exigente, fazendo com que a qualidade do software comercializado aumente cada vez mais. Para atingir esse nível de excelência, o desenvolvimento da aplicação não pode ser arbitrário. É necessário alguma técnica que possibilite a identificação de falhas do software.

Um destes exemplos de validação dos produtos são os testes de software. Segundo MYERS [1], teste de software é o processo de executar um programa ou sistema com a intenção de encontrar erros. A partir deste processo, revelam-se falhas de um determinado produto e essas tendem a ser identificadas e corrigidas pelo time de desenvolvimento da empresa em tempo hábil para entrega.

Embora eficientes, os testes devem seguir uma metodologia que garanta sua efetividade. O que não acontece em boa parte das empresas de pequeno e médio porte, já que estas efetuam os testes de forma informal, ou seja, não seguem uma sistematização. Como os sistemas informatizados são constantemente alterados e melhorados, fica difícil manter a qualidade do produto fazendo com que o resultado final nem sempre corresponda ao esperado.

Os testes podem ser manuais e/ou realizados via ferramentas de automação de teste. Ambas as abordagens são complementares e apontam possíveis falhas e potenciais melhorias no software. Segundo TOGNOLO [4], o teste de software é o mecanismo essencial para termos segurança no código. Ele é uma técnica de verificação para descoberta de erros em sistemas. No entanto, o teste pode ser muito dispendioso em tempo e recursos caso seja necessário analisar o software em sua totalidade, para isso pode-se contar com ferramentas para automação dos testes, contribuindo com produtividade da equipe de testes. Independente de ser realizado individualmente ou em grupos, o teste de software sempre está sujeito a falhas.

As metodologias ágeis trouxeram um grande diferencial aos testes de software. Segundo STEFFEN [11], essa é uma nova forma de gestão e desenvolvimento de Software que usa uma abordagem de planejamento e execução interativa e incremental voltado para processos empíricos. Diante das metodologias ágeis se derivou o *Test Driven Development* (TDD), que é uma técnica de teste cujo processo é cíclico para cada nova funcionalidade a ser desenvolvida. O TDD será a técnica utilizada neste trabalho, que se apresenta como uma proposta.

Este artigo apresenta uma proposta de um processo de teste de software para uma empresa de desenvolvimento de software da cidade de Frederico Westphalen – RS. Propõe- se uma metodologia de testes que se enquadre ao processo de desenvolvimento atual da empresa.

Atualmente, o processo de teste é feito de forma informal na empresa objeto deste estudo. Com a proposta de uma metodologia ágil de teste de software, espera-se que haja uma evolução positiva nos processos internos de desenvolvimento de software e a consequente melhora na qualidade dos produtos gerados.

As principais contribuições desse trabalho são:

- Formalizar processo atual de desenvolvimento de software da empresa localizada na cidade de Frederico Westphalen;
- Propor um processo de testes proposto para a empresa;
- A inclusão da automação de teste no processo proposto;

Este artigo está estruturado da seguinte forma: a Seção II apresenta os conceitos de teste de software; a Seção III apresenta os conceitos de *Test Driven Driven Developmen;*. a Seção IV o contexto em que o artigo está baseado; a Seção V apresentará o planejamento do TDD; a Seção VI apresenta a criação dos testes baseados na metodologia TDD; a Seção VII apresenta o planejamento dos testes de regressão; a Seção VIII apresenta a automação do processo de teste; a Secão IX apresenta as lições aprendidas no decorrer do projeto; a Seção X apresenta os trabalhos relacionados. Por fim, são apresentadas as conclusões e referências bibliográficas utilizadas nesse trabalho.

II. TESTE DE SOFTWARE

O desenvolvimento de software não é uma tarefa simples. Pelo contrário, pode se tornar bastante complexa dependendo das características e dimensões do sistema a ser criado. Por isso, está sujeita a diversos tipos de problemas que acabam resultando na obtenção de um produto diferente daquele que se esperava [5].

Diversos fatores podem contribuir na geração de problemas no desenvolvimento de software. Para identificar estas falhas, é preciso valer-se de técnicas de observação deste. Uma forma para se observar o software em funcionamento é avaliar o comportamento do mesmo, ou seja, se está funcionando de forma a suprir as necessidades especificadas pelo cliente. Visando verificar o grau de qualidade que o software se encontra, uma das formas de se fazer esse processo é através de teste de software.

Segundo BARTIÉ [18], os erros ocorrem em todo o processo de desenvolvimento de um sistema, porém, a maior incidência dos erros está concentrada nas fases iniciais, devido à má especificação e entendimento dos objetivos a serem alcançados.

Teste de software é uma técnica de verificação que começou a ser utilizado informalmente na década de setenta como um passo natural, com os próprios desenvolvedores fazendo os testes como forma de correção a erros de execução. Buscando maior efetividade, no decorrer do tempo surgiram novas metodologias de teste. Elas estão em constante evolução, pois são tratadas como um processo relevante garantindo aumento da qualidade e funcionalidade do software, tornando o processo cada vez mais útil.

Devido aos altos custos de manutenibilidade, as empresas investem cada vez mais na qualidade de seus sistemas antes de os mesmos entrarem em funcionamento comercialmente. Frequentemente, gasta-se muito tempo e dinheiro em testes e na correção dos erros encontrados. Teste de software é por si só, uma atividade muito dispendiosa e o custo de não testar é potencialmente muito maior [17].

O objetivo do teste é avaliar o comportamento do software baseado no que foi especificado no pedido do cliente. Existem várias definições para teste de software:

• Processo de executar um programa ou sistema com a intenção de encontrar erros [1].

• Qualquer atividade que a partir da avaliação de um atributo ou capacidade de um programa ou sistema seja possível determinar se ele alcança os resultados desejados [2].

• Teste de software é uma técnica dinâmica usada par certificar que o software está de acordo com suas especificações [3].

Com a exigência cada vez maior do mercado consumidor, é imprescindível que se garanta a qualidade nos produtos de software comercializados no mercado. Diante dessa necessidade muitas empresas de desenvolvimento têm aderido ao teste de software.

Sendo assim, os testes se tornaram parte indissociável do desenvolvimento de software. Atualmente existem várias técnicas de teste, sendo duas abordagens conhecidas a de caixa branca e caixa preta.

Teste de caixa branca ou teste estrutural são baseados no código fonte do software. Nele são testados os caminhos lógicos do software. O engenheiro de sistema realiza o teste direto no código fonte do software. Segundo PEZZE [9] a estrutura do software em si é uma fonte valiosa de informação para a seleção de casos de teste e determinar se um conjunto de casos de teste tem sido suficientemente completo.

Teste caixa preta ou funcional são realizados em cima dos requisitos funcionais do software, ou seja, na descrição das funções do mesmo. Como o testador não tem conhecimento do código fonte do software, o foco fica nas funções que o programa deve realizar. Segundo PEZZE [10], casos de teste de caixa preta devem começar com o processo de especificações dos requisitos do software e continuar através de cada nível.

Este método tenta assegurar que as funções do sistema de aplicação atinjam os objetivos esperados e de forma correta. O teste de caixa preta concentra-se nos requisitos funcionais do software. Através dele, torna-se possível verificar as entradas e saídas de cada unidade [19]. Ambos os estilos de testes podem ser manuais ou automatizados.

Testes de software manuais e automatizados trabalham em completude, porém existem algumas peculiaridades entre eles, que - quando usadas de maneira inteligente – contribuem para a agilidade e eficiência dos testes. Por exemplo: os testes manuais são ideais para serem usados em tarefas que exijam mais pensamento lógico do usuário, pois recebem atenção pontual do testador. Já os testes automatizados são ideais para serem aplicados em testes repetitivos, que não necessitam de tanto foco por serem aplicados em situações semelhantes a serem resolvidas.

Segundo CRESPO [20] deve ser ressaltado que a atividade de teste exige conhecimento, planejamento, projeto, execução, acompanhamento, recursos e também uma grande interação com as outas equipes.

III. TEST DRIVEN DEVELOPMENT (TDD)

O TDD criado por KENT [6], criador do TDD, não foi uma ideia totalmente original, o processo do TDD foi baseado em livros do seu pai que também era programador.

A ideia do TDD é simples: o desenvolvedor deve começar pelo teste e fazer com que seu código passe por ele. Os princípios fundamentais do TDD que temos que adotar na sua implantação é:

- Sempre escrever o teste antes de escrever o código;
- Sempre escrever um código simples para passar no teste realizado anteriormente;
- Escrever testes pequenos, para que possa testar uma quantidade maior de teste;
- Sempre escrever testes rápidos, ou seja, a execução não deve demorar muito tempo.

A Figura 1 apresenta o ciclo de vida do TDD. O ciclo deve ser seguido à risca. Como a mostra a Figura 1: 1º passo: o desenvolvedor deve escrever o teste mais simples possível, 2º passo: obrigatóriamente o teste falhará. 3º passo: o testador escreve um teste que falhará no teste. 4º passo: é realizada a refatoração do código, removendo duplicidades de dados e algum código que não seja usado.

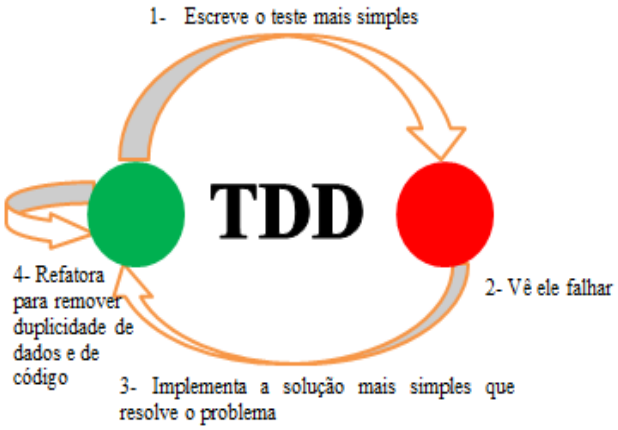

Figura 1 – Clico de Vida do TDD [24]

Segundo ANICHE [8] muitos desenvolvedores afirmam que executar esse ciclo pode ser vantajoso para o processo de desenvolvimento. Algumas vantagens são:

- **Foco no teste e não na implementação:** Começando pelo teste o desenvolvedor pensa somente no que a classe deve fazer esquecendo momentaneamente da implementação, isso facilita pensar em vários cenários de teste para a classe que está em desenvolvimento.
- **Código nasce testado:** se o desenvolvedor realiza o ciclo corretamente, isso indica em que todo o código escrito tenha ao menos um teste de unidade, para que possa ser verificado que o mesmo funciona corretamente.
- **Simplicidade:** o desenvolvedor deve tentar fugir de soluções complexas, o mesmo deve sempre tentar realizar a busca pelo código mais simples constantemente. O desenvolvedor que pratica o TDD deve escrever o código que resolva os problemas que estão sendo apresentados pelo teste.
- **Melhor reflexão sobre o design da classe:** no ambiente tradicional de desenvolvimento, acontece de várias vezes a falta de coesão ou excesso de acoplamento que o desenvolvedor acaba causando, pois o mesmo acaba pensando mais na implementação esquecendo-se de como a classe realmente deve funcionar no processo. Começando pelo teste, o programador pensa como a classe deverá funcionar junto com as demais classes do projeto.

Segundo LOPES [7], quando entregamos algo com TDD, estamos seguros de que estamos entregando a coisa certa com um risco menor. Ao executar os passos do TDD com êxito estamos assegurando que o processo está sendo feito com qualidade.

Na empresa objeto deste estudo, atualmente o processo do TDD não é praticado, o que motivou a inserção do mesmo como uma forma de garantir a integridade do software gerando mais qualidade ao que está sendo implementado. O método de testes informal realizado até agora é feito na totalidade do software, depois de ser desenvolvido. Também por este motivo a implementação do TDD se torna indicável, já que aborda o software em partes, garantindo mais agilidade e menor chance de erros. Por ser um software em produção e pronto, o TDD é aplicado nas novas modificações e novas funcionalidades que estão sendo desenvolvidas no sistema.

O grande intuito é que a forma como o software é testado hoje, de forma informal, sofra uma grande diferença com o TDD. Assim sua qualidade e produtividade aumentará e se por algum motivo seja feita alguma alteração nas funcionalidades desenvolvidas com o TDD, os testes já estão prontos pois é só executá-los novamente – o que garante a qualidade já empregada antes no desenvolvimento do software.

IV. CONTEXTO E PROPOSTA

A empresa, no qual será realizado o estudo de caso deste trabalho, atua há vinte e três anos no ramo de tecnologia com ênfase no desenvolvimento de sistemas para gestão empresarial. Atualmente, conta com quinze funcionários e cerca de quinhentos clientes ativos. Os softwares desenvolvidos pela empresa atendem a toda região sul e parte do nordeste do Brasil. Os produtos desenvolvidos são voltados para estabelecimentos comerciais que contemplam produtos e serviços. Alguns dos ramos atendidos são: farmácias, drogarias, supermercados, indústrias, distribuidoras de medicamentos, distribuidora de bebidas, postos de combustíveis, entre outros.

A Figura 2 apresenta o processo utilizado atualmente pela empresa. O processo é utilizado para cada novo requisito ou alteração de requisito solicitado pelos clientes. O desenvolvimento desses produtos ocorre da seguinte maneira:

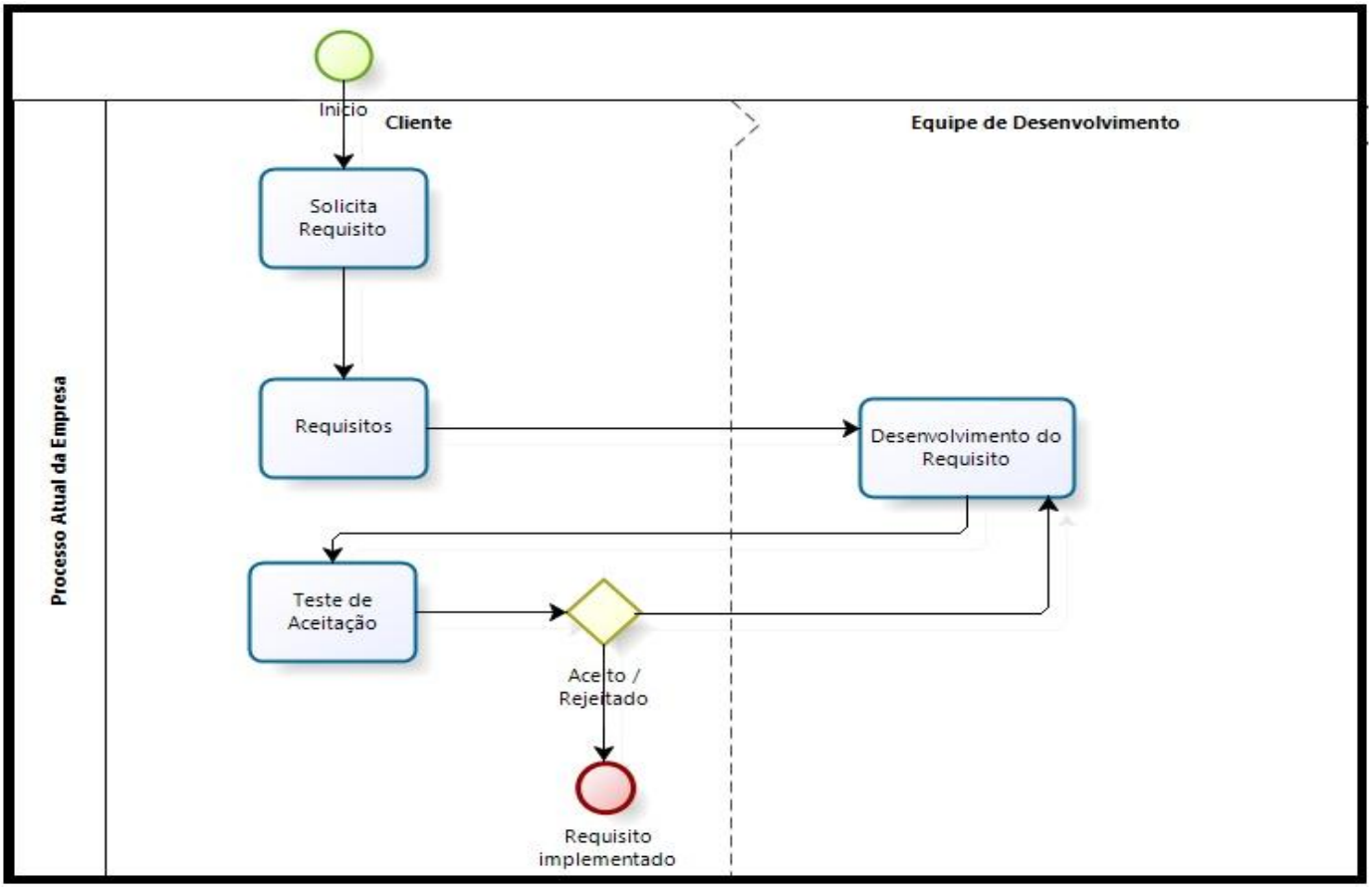

Figura 2: Processo utilizado atualmente pela empresa

• **Cliente**: ator responsável pela solicitação de um novo requisito ou alguma mudança de requisitos.

• **Solicita Requisito**: processo que é iniciado pelo cliente por e-mail ou telefone. Representa a solicitação de um novo/mudança de um novo requisito.

• **Requisitos**: artefato gerado pela solicitação do cliente via telefone ou e-mail e é encaminhado ao time de desenvolvimento.

• **Equipe de Desenvolvimento**: atores responsáveis por desenvolver os requisitos.

• **Desenvolvimento de Requisito:** processo realizado pelo time de desenvolvimento que inclui análise, implementação e teste informal dos requisitos.

• **Teste de Aceitação**: Este processo o cliente valida o requisito solicitado. Caso o requisito não esteja de acordo, ele é rejeitado e a solicitação volta para o desenvolvimento do requisito. Caso o requisito esteja de acordo, ele é aceito para implantação.

Na empresa à qual é realizado o estudo de caso é usada a linguagem de desenvolvimento Delphi. A linguagem possui um ambiente de desenvolvimento integrado onde é possível editar código, testar a aplicação, verificar erros além de compilar a aplicação.É possivel visualizar como é o processo da proposta deste trabalho de graduação em sistemas de informação através da Figura 3, onde é visto que:

Empresa: possui processo informação de teste de software, a cultura organizacional da empresa não favorece este processo de teste, existe uma grande demanda de modificações e implementação de novas aplicações e melhorias.

Processo de teste: a produtividade será muito maior assim como a qualidade do software produzido, usando o processo de testes do TDD e testes de regressão.

Automação: os testes de regressão serão automatizados via DUnit.

Produto Final: por fim teremos um produto final com qualidade e testado, garantindo a satisfação do cliente e tendo um processo de teste e desenvolvimento de software estruturado.

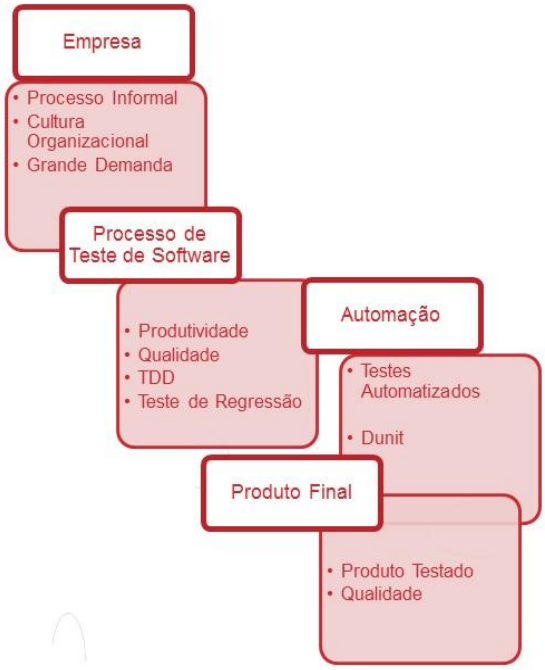

Figura 3: Proposta do TGSI

V. PLANEJAMENTO DO TDD

A grande mudança de regras de negócio existente no meio empresarial, exige que as empresas, sejam eficientes e criativas para poder acompanhar a evolução das exigências de seus clientes com êxito. Para garantir a qualidade de um software (que funcione de forma correta e atenda às necessidades de organização do cliente) deve-se implantar um processo de testes eficiente, ou seja, iniciar o projeto pelos seus testes, fazendo com que todo o desenvolvimento do código do programador nasça testado, é isso que prega a metodologia TDD.

O princípio básico do TDD é primeiro o teste e depois a implementação. Desta forma esta técnica garante que sua aplicação tenha código suficiente para suprir a demanda, e nada mais. É uma técnica simples que pode elevar o nível das aplicações presentes atualmente na empresa objeto deste estudo.

A Figura 4 apresenta o processo proposto no estudo de caso na empresa. O processo poderá ser usado para cada novo requisito ou alteração de requisito solicitado pelos clientes. Foram introduzidos quatro rotinas no processo atual, realizando o processo do TDD, estas quatro rotinas são:

- **Testador:** ator responsável pelos testes da aplicação, seja ela uma implementação nova ou uma mudança na aplicação.
- **Desenvolvimento dos Testes:** desenvolvimento dos testes usando a metodologia TDD, realizados pelo Testador.
- **Execução dos Testes:** execução dos testes em TDD pelo testador, aplicando a metodologia.
- **Refatoração do Código Desenvolvido:** é feita a refatoração do código desenvolvido pelo time de desenvolvimento.

O processo proposto pela Figura 4, mostra–se prático para ser aplicado na atual estrutura da empresa, pois é eficiente no consumo de tempo e energia dispendidos no roteiro do teste.

VI. CRIAÇÃO DOS TESTES DO TDD

A grande vantagem da metodologia TDD é que seus testes são realizados antes do código ser implementado. A ideia é basear-se na descrição da funcionalidade para escrever os testes antes de implementá-los, com isso a funcionalidade torna se mais clara e supostas dúvidas sobre o funcionamento correto já são sanadas antes que a funcionalidade entre em desenvolvimento. Nesta seção vamos abordar como os testes são criados na abordagem TDD.

A Tabela 1 mostra a funcionalidade a ser aplicada nos testes em TDD. Separada em três tópicos: nome, objetivo e descrição e regras de negócio, a tabela serve para separar e deixar mais clara a realização dos testes pelo testador.

No processo atual foi desenvolvido um conjunto de casos de teste para aplicação no desenvolvimento da ferramenta, como por exemplo, o caso de teste 01.

Na atual aplicação foi selecionada uma nova funcionalidade para darmos inicio aos testes via TDD. A Tabela 1 apresenta a funcionalidade a ser implementada:

Funcionalidade 1
Nome: Controle de Tempo por Atividade
Objetivo: A funcionalidade tem por objetivo controlar o tempo dos funcionários numa empresa que presta serviços.
Descrição e regras de negócio: O principal objetivo desta funcionalidade é o controle de tempo de trabalho dos funcionários de uma empresa relacionado à carga de trabalho do mesmo. Uma forma de fornecer informações ao administrador que demonstrem as estatísticas de horas cumpridas e de trabalho realizado por cada um de seus colaboradores. Uma ferramenta para medir a eficiência de cada funcionário. Por meio de relatórios dos serviços prestados por funcionários, poderá medir quanto tempo o mesmo levou para realizar o trabalho solicitado; a que horas começou e que horas terminou. O sistema trabalhará com o sistema de 24 horas, não se limitando apenas ao horário comercial.

Tabela 1: Funcionalidade a ser implementada

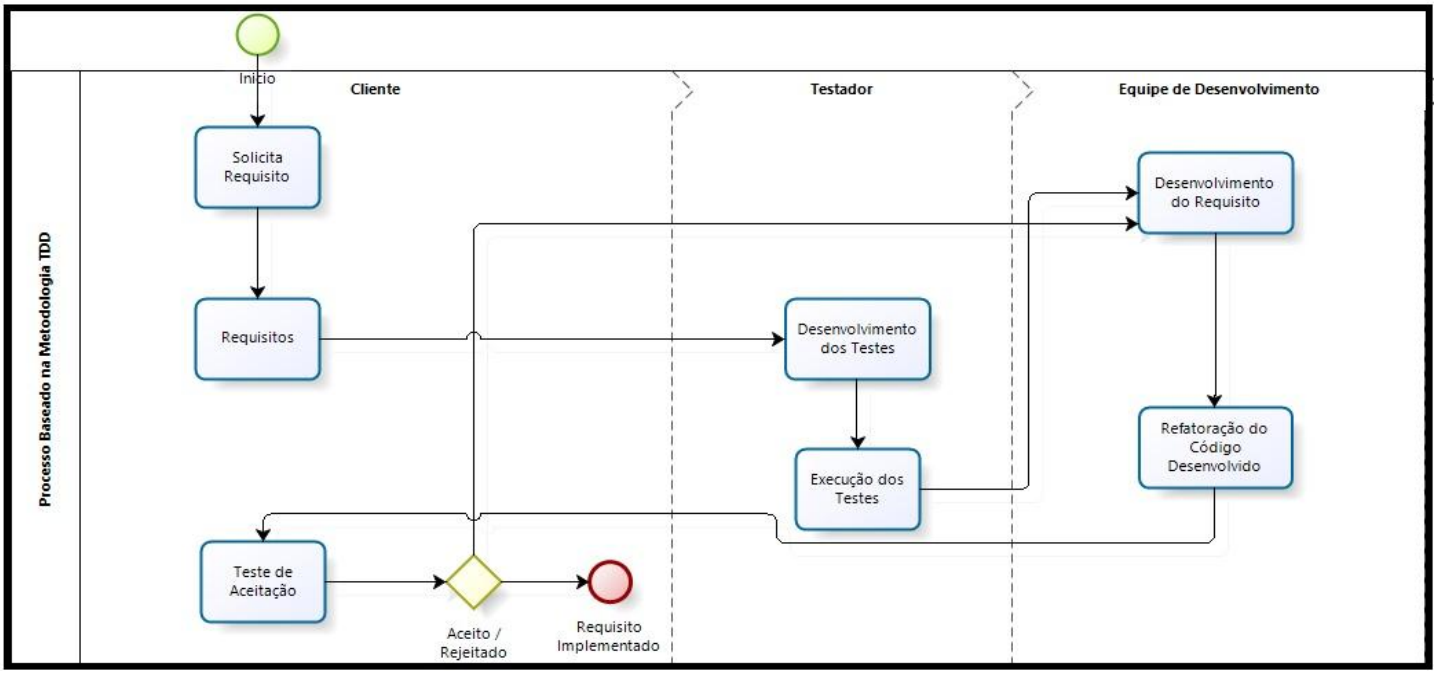

Figura 4: Processo baseado na metodologia TDD

Caso de Teste 01
Nome: Controle de tempo por atividade
Objetivo: Inserir hora de funcionário
Passos: 1: Usuário está na tela de Serviços 2: Usuário vai na aba controle de tempo 3: Usuário clica com o botão direito do mouse e seleciona a opção, novo controle de tempo 4: Sistema abre a tela de cadastrado de horas por funcionário 5: Usuário insere o funcionário x 6: Usuário insere-o no campo Motivo: o trabalho a ser efetuado pelo funcionário 7: Usuário insere no campo tempo, a hora de início da atividade. 8: Usuário insere o valor da hora trabalhada desse funcionário nesse determinado serviço. 9: Usuário clica em OK. 10: Sistema valida as informações e insere os dados na tela de serviço.
Resultado Esperado: 1: Sistema irá inserir as informações com opção de edição e de inserir novos funcionários no serviço, a fim de contabilizar as horas trabalhadas e para fins de preço final do serviço desempenhado.

Tabela 2: Caso de Teste: Nova Funcionalidade

Como é possível observar, o caso de teste da Tabela 2, exemplifica o uso normal da funcionalidade a ser desempenhada pelo usuário do sistema, a fim de poder adicionar, editar, contabilizar as horas e o custo de cada funcionário para cada de serviço, tendo a opção de medir a produtividade dos funcionários. Com esta função podem ser criados diversos outros casos de testes, onde é possivel explorar de outras formas o requisito, podendo haver testes exploratórios, testar a integração do requisito, testar os limites, sendo que cada caso tem sua particularidade e pode depender de algum fator variável dentro da aplicação atual.

VII. TESTE DE REGRESSÃO

O processo de teste de software realizado no produto tem a principal função de determinar se o mesmo atingiu as especificações conforme o ambiente projetado.

Os testes de regressão tem objetivo de garantir que o sistema não tenha nenhum defeito após alguma modificação realizada, tendo como referência o código original (anterior à modificação). Estes testes são aplicados após alguma correção de defeito ou quando uma nova funcionalidade é implementada.

Este tipo de teste é realizado em sistemas legados. Dependendo do tamanho da aplicação, podem ser usadas algumas ferramentas de automação de teste, pois pode se poupar tempo usando uma aplicação para executar esses testes. Os testes de regressão tem grande vantagem na empresa objeto de estudo deste trabalho. O grande fluxo de atualizações realizadas nos softwares da empresa, é necessário que a equipe tenha alguns conjuntos de testes de regressão a fim de rodá-los a cada atualização fechada. Como são implementadas mudanças nos produtos a cada nova versão, é necessário ter esse padrão de qualidade em seus produtos.

A Figura 5 apresenta o processo utilizado no planejamento dos testes de regressão na empresa. O desenvolvimento desses produtos ocorre da seguinte maneira:

- **Cliente**: ator responsável pela solicitação de um novo requisito ou alguma mudança de requisitos.
- **Solicita Requisito**: processo que é iniciado pelo cliente por e-mail ou telefone. Representa a solicitação de um novo/mudança de um novo requisito.
- **Requisitos**: artefato gerado pela solicitação do cliente via telefone ou e-mail e é encaminhado ao time de desenvolvimento.
- **Time de Desenvolvimento**: atores responsáveis por desenvolver os requisitos.
- **Desenvolvimento de Requisito:** processo realizado pelo time de desenvolvimento que inclui análise, implementação e teste informal dos requisitos.

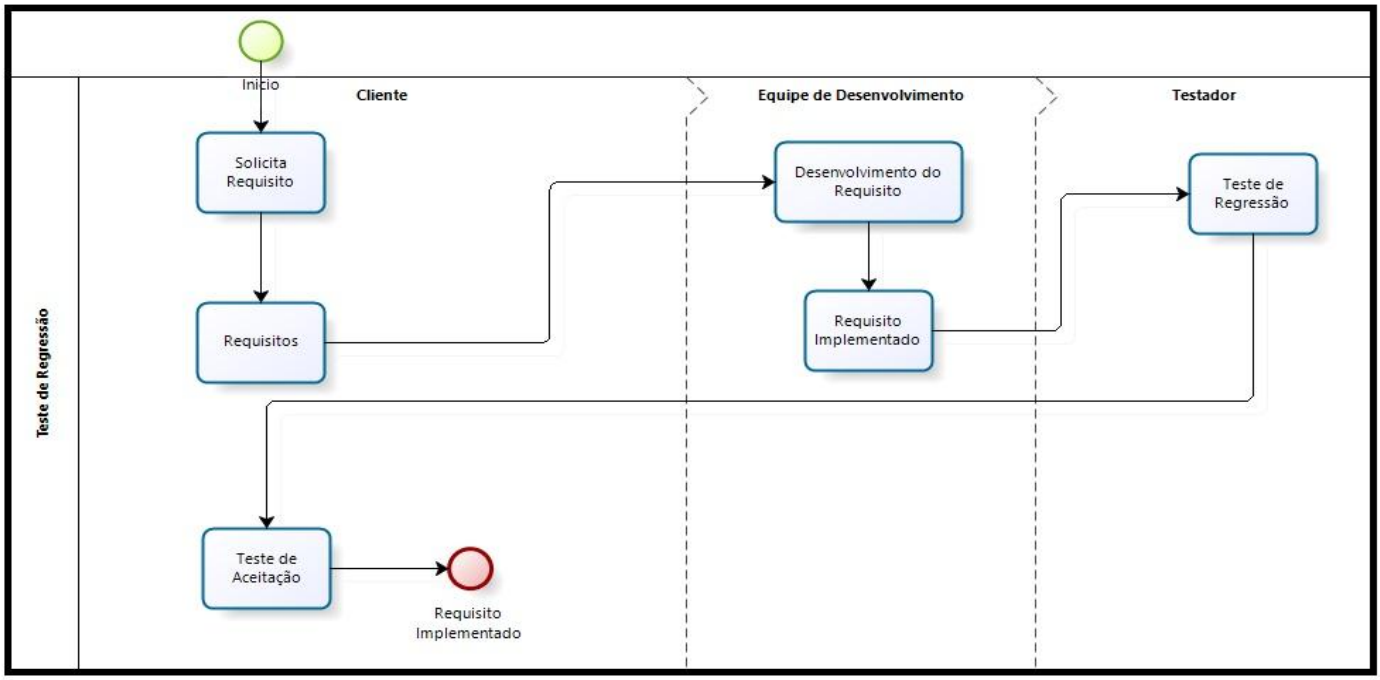

Figura 5: Teste de Regressão

• **Requisito implementado**: processo final no qual o requisito já foi validado e está pronto para implementação no cliente.

Um exemplo de teste de regressão que a Figura 5 mostra é aplicada em cima de uma funcionalidade muito importante no sistema atual a ser testado.

• **Teste de Regressão:** os testes de regressão são executados quando a implementação está pronta.

• **Teste de Aceitação**: processo no qual o cliente valida o requisito solicitado. Neste processo o cliente valida o requisito solicitado. Caso o requisito esteja de acordo, ele é aceito para implantação.

Foi criado um conjunto de testes de regressão com um total de 62 casos de testes, que foram executados na aplicação, como mostra a Tabela 3:

Testes de Regressão	
Testes Positivos:	42
Testes Negativos:	20
Total:	62

Tabela 3 – Testes de Regressão

Neste conjunto de teste foram realizados um total de 62 casos de testes e todos executados. Estes podem ser definidos a partir dos requisitos, podendo ser do tipo negativo (ações imprevistas) e positivo (ações previstas) [16], sendo que deste montante de 62 casos, 42 eram casos de teste positivos e 20 eram casos de teste negativos.

Os testes de regressão são executados após a implementação já estar pronta, logo se torna comum que não haja tempo suficiente para realização dos testes de regressão na aplicação. Neste caso tendo um conjunto de testes de regressão pronto, diminui o tempo e melhora a produtividade da equipe perante os testes.

A Tabela 4 mostra um exemplo de caso de teste de regressão positivo, onde o usuário tem a função de cadastrar um novo serviço, conforme o cliente deseja.

Caso de Teste 02
Nome: Novo serviço
Objetivo: Cadastrar novo serviço
Passos: 1: Usuário vai na opção Serviços 2: Usuário abre um novo serviço 3: Sistema solicita que escolha um cliente 4: Usuário escolhe um cliente e preenche os campos em marrons que são obrigatórios (tais campos são: cliente, motivo, identificação, técnico responsável, motivo, condição de pagamento) 5: Usuário clica em salvar serviço 6: Sistema salva o serviço e deixa com Status Aguardando avalição técnica.
Resultado Esperado: 1: Serviço se encontra aberto para o técnico responsável começar o processo.

Tabela 4: Caso de Teste de Regressão

No caso de teste acima, temos uma situação simples do dia a dia de funcionamento do sistema, solicitado que fosse realizado um novo serviço. Conforme o resultado esperado o técnico responsável pelo serviço entrará no sistema e ver seus serviços pendentes, nele constará este novo serviço aberto. Lá ele poderá inserir produtos usados para efetuar o serviço, bem como a troca de status do serviço, podendo alterar para outros diversos status como: Em produção, Aguardando aprovação do cliente, Pronto, Saiu para entrega, Sem conserto, Serviço finalizado.

Neste contraponto temos um teste negativo que baseia-se em o usuário tentar finalizar o serviço sem laudo técnico, ou seja, sem um motivo final para ser apresentado pelo funcionário ao cliente no final do serviço.

Caso de Teste 03
Nome: Finalizar Serviço
Objetivo: Finalizar serviço sem colocar o Laudo
Passos: 1: Usuário vai na opção Serviços 2: Usuário localiza o serviço aberto 3: Sistema abre a tela de serviço com seu atual serviço na tela 4: Usuário preenche os dados do serviço e não preenche o laudo tecnico. 5: Usuário clica em finalizar o serviço. 6: Sistema finaliza o serviço e deixa com serviço finalizado.
Resultado Esperado: 1: Serviço se encontra finalizado para o técnico responsável

Tabela 5: Caso de Teste de Regressão Negativo

Neste caso de teste de regressão negativo, o resultado esperado era que o sistema permitisse que fosse finalizado o processo sem ter um laudo técnico, porém, atualmente o processo de inserção do laudo técnico é obrigatório não permitindo o fechamento da ordem de serviço sem o mesmo estar informado. Ou seja, a função do caso de teste 03, não é válida conforme o esperado pela aplicação.

VIII. AUTOMAÇÃO

A automação permite executar uma maior quantidade de testes em um tempo relativamente muito menor. A automação de testes consegue trazer algumas vantagens para o processo de desenvolvimento, como por exemplo: chance de erro humano é minimizada, redução do esforço dos testadores.

Segundo CAETANO [13] a automação de testes, é a uma boa prática otimizada e maximizada de testar o software. A necessidade de automatizar os testes virá naturalmente como resultado da evolução da maturidade do processo de testes.

Este processo de automação se dá com o uso de ferramentas específicas para estes processos, tendo o objetivo de avaliar se o requisito está sendo desenvolvido conforme o esperado. Para a realização de testes automatizados, existem ferramentas prontas que auxiliam na criação, desenvolvimento e manutenção desses testes. Elas permitem que scripts com testes automatizados sejam criados, alterados e executados" [21]. A ferramenta escolhida para execução do processo foi a DUnit.

DUnit é um framework de teste unitário para programas Borland Delphi. Pode-se verificar a interface do framework na Figura 6. Ele foi originalmente inspirado no framework JUnit escrito em Java por Kent Beck e Erich Gamma, mas evoluiu

para uma ferramenta que usa muito mais do potencial do Delphi para ser muito mais útil para desenvolvedores Delphi [12]. Com o framework DUnit, podemos verificar se cada método de uma classe funciona conforme o esperado. Uma grande vantagem do DUnit é que o mesmo pode ser executando em qualquer conjunto de testes. A automação de testes será usado para testes de regressão no sistema legado.

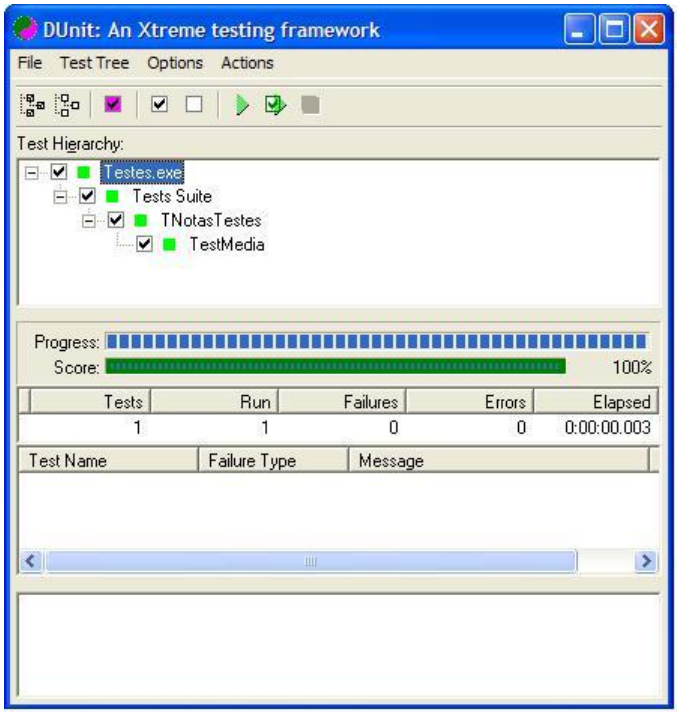

Figura 6 – DUnit: Ferramenta de Teste Automatizado

A partir de uma aplicação artificial desenvolvida apenas para que possamos mostrar o funcionamento do DUnit, foi possível visualizar o framework DUnit em seu total funcionamento. A Tabela 5 mostra o caso de teste da aplicação desenvolvida para o teste estrutural via Dunit.

Caso de Teste
Nome: Teste de Software Dunit
Objetivo: Verificar Resultado esperado
Passos: 1: Usuário abre a aplicação 2: Usuário insere número no campo Nun1 3: Usuário insere número no campo Nun2 4: Usuário insere número no campo Valor esperado 5: Usuário clica em Realizar Testes (Somar/Subtrair) 6: Sistema abre o framework do Dunit. 7: Sistema executa os testes conforme o desejado.
Resultado Esperado: 1: Dunit testará a estrutura da aplicação

Tabela 5: Caso de Teste de Regressão

Na Figura 7 pode-se visualizar a tela principal da aplicação artificial, tendo o objetivo de testar o resultado esperado inserido pelo usuário. Através do DUnit pode-se realizar testes de forma bem simples, testando a estrutura e o resultado da aplicação, podendo optar por dois testes de software:

- Teste da soma: aplicação irá verificar os parâmetros inseridos nos campos e realizará o teste, testando se o resultado está conforme o esperado.
- Teste Subtração: da mesma forma que o teste anterior, porém realizando o inverso, da operação, testando os parâmetros inseridos nos campos para ver se o resultado está de acordo com a subtração.

Como é possivel visualizar a aplicação consite com apenas um botão chamado “Realizar Testes(Somar / Subtrair)”, este botão tem a função de chamar o framework DUnit, onde poderão ser realizados os testes, chamados de TestSoma e TestSubtracao.

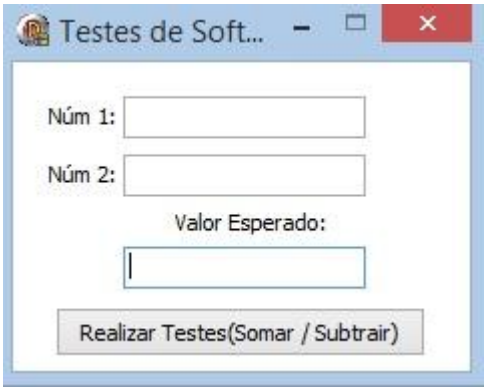

Figura 7 – Aplicação de Teste

Como podemos exemplificar abaixo com a Figura 8 podemos visualizar um teste de software positivo:

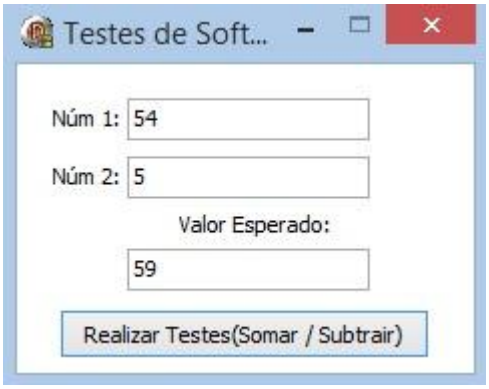

Figura 8 – Teste que passou.

Visualizando a aplicação da Figura 8, podemos ver que o primeiro parâmetro inserido é o 54 e o segundo parâmetro é o 5, o terceiro parâmetro é o valor que o usuário espera que sua operação tenha como resultado, neste caso é 59, assim ao clicar no botão Realizar Testes (Somar / Subtrair), a aplicação chama o framework Dunit, responsável por toda operação de testes estruturais da aplicação.

A Figura 9 apresenta o TestSoma, que testa se o valor esperado inserido na aplicação da Figura 8 está de acordo com o TestSoma, executado pelo framework, Dunit, caso esteja de acordo, o teste passa, mostrando um grid com seus números, por exemplo ao executarmos o TestSoma, o grid mostra, que ocorreu:

- 1 teste

- 1 execução
- 0 Falhas
- 0 Erros
- Tempo de teste de 12 milésimos

Na Figura 9, temos todas as informações necessárias para sabermos se o teste está de acordo ou não com o que estamos esperando. Uma parte do código do teste via DUnit, é apresentado na Figura 10. É possível visualizar que a partir da linha 46 do código pode-se ver a primeira *Procedure* do teste da aplicação desenvolvida, a primeira Procedure chama-se SetUp, essa será chamado antes de cada teste ser executado, para que permita o usuário preparar o teste com as informações necessárias antes de sua execução, a Procedure chamada TearDown é chamada pela aplicação depois que o teste é realizada, sua função é liberar os recursos alocados pelo teste, a partir da linha 56 temos as *Procedure* do TestSoma (nome dado ao teste para conferir se o valor final é resultado de uma soma) e na linha 61 temos a *Procedure* do TestSubtracao (nome dado ao teste para conferir se o valor final é resultado de uma subtração). O DUnit permite que códigos simples como o presente na Figura 10, possibilitem que seja realizado o teste de unidade de forma correta.

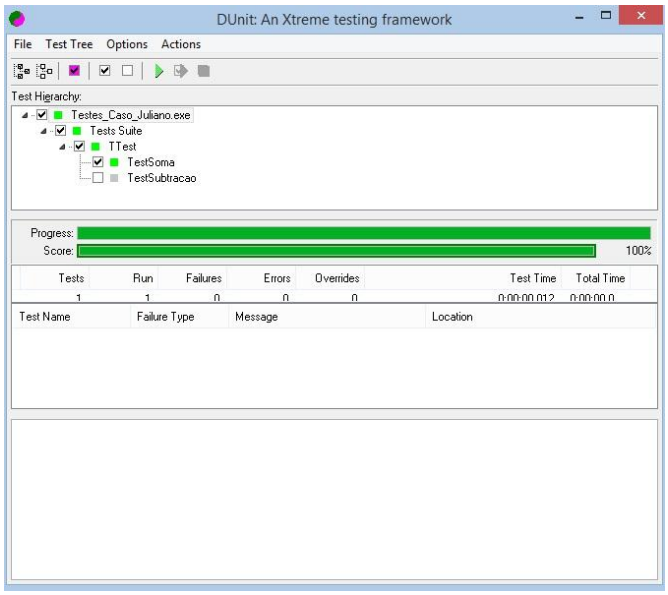

Figura 9 – Teste em Dunit que passou

```
43
44  { TTest }
45
46  procedure TTest.SetUp;
47  begin
48    inherited;
49  end;
50
51  procedure TTest.TearDown;
52  begin
53    inherited;
54  end;
55
56  procedure TTest.TestSoma;
57  begin
58    CheckEquals(StrToInt(FrmPrincipal.EdtVlEsperado.Text),Calc.Soma
59    (StrToInt(FrmPrincipal.EdtN1.Text),StrToInt(FrmPrincipal.EdtN2.Text)));
60  end;
61  procedure TTest.TestSubtracao;
62  begin
63    CheckEquals(StrToInt(FrmPrincipal.EdtVlEsperado.Text),Calc.Sub
64    (StrToInt(FrmPrincipal.EdtN1.Text),StrToInt(FrmPrincipal.EdtN2.Text)));
65  end;
66
67  procedure TFrmPrincipal.BtnSomaClick(Sender: TObject);
68  begin
69    GUITestRunner.RunRegisteredTests;
70  end;
71
72  initialization
73    TestFramework.RegisterTest('Tests Suite', TTest.Suite);
```

Figura 10 – Código TestSoma/Subtração

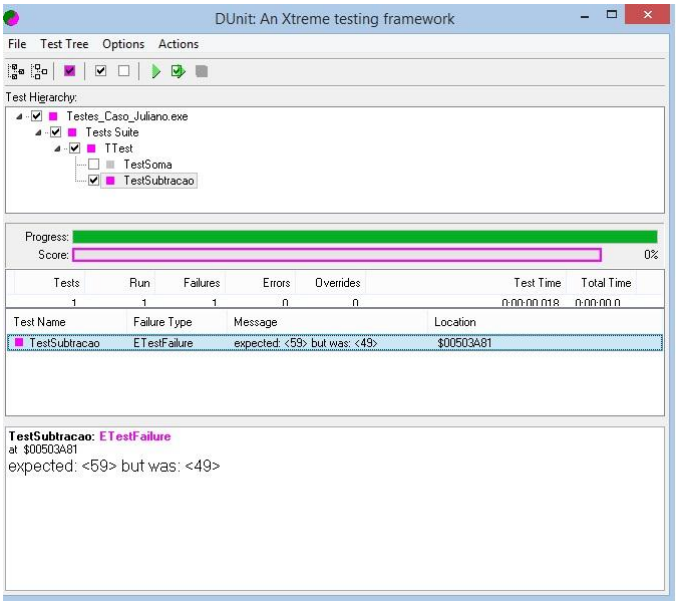

Figura 11 – Teste que não passou

Neste teste de unidade executado via Dunit, são inseridos os mesmos parâmetros do teste positivo, entretanto executamos este segundo teste aplicando o TestSubtração, que tem a função de verificar se o primeiro parametro (num1) e o segundo parametro (num2). Subtraídos um do outro dão o resultado esperado, sendo assim, após a execução pode-se verificar que o teste encontrou uma falha. Ao executarmos o

TestSubtração o valor esperado seria 49 e não 59 como o resultado.

Os números informados pelo DUnir referente a execução deste teste são:

- 1 Teste
- 1 Execução
- 1 Falha
- 0 Erros
- Tempo do Teste 18 milésimos

Conforme a falha do TestSubtração, o DUnit traz o resultado e o por que da crítica na interface da aplicação, dependendo do tamanho da aplicação é a crítica, neste caso foi mostrado a seguinte mensagem de falha no teste, na Figura 11. Este teste acaba por ser um teste negativo, pois o mesmo é executado com sucesso e passa pelo teste, porém o resultado é diferente do esperado.

IX. Lições Aprendidas

No decorrer deste trabalho aplicação do TDD apresentou dificuldades. Por exemplo, cultura organizacional não possuindo uma estruturação para o desenvolvimento de testes. Isso ocorreu pela empresa já ter processos usados por muito tempo. A mundança desses processos requer não apenas uma mudança organizacional mas também uma mudança de estrutura em termos de recursos. É por isso que qualquer um que deseje inserir o processo de testes em uma organização deve ter claro qual seu objetivo e os resultados pretendidos com essa ação.

Observou – se que a introdução de um processo de testes mesmo simples pode minimizar os riscos e diminui o tempo gasto em correções. Pela característica do TDD, o código já nasce testado e isso significa que caso haja algum erro no código, ele será identificado logo no começo do desenvolviment. Dessa maneira, a possibilidade de o desenvolvedor ter que refatorar o código que escreveu a algum tempo é minimizada. Além disso possibilita a padronização do código.

A aplicação de um processo testes de software deve começar pelas noções mais rudimentares. O TDD precisa estar claro para os desenvolvedores e os mesmos devem segui – lo.

Observou-se que num primeiro momento, desenvolvedores de software podem ter dificuldades na implantação do TDD, mas obtêm vantagens significativas com a implantação do TDD a médio e longo prazo.

X. Trabalhos Relacionados

Nessa seção são apresentado alguns dos trabalhos relacionados. Os trabalhos abordam estudo de caso utilizando metodologias ágeis, principalmente o TDD.

FRONZATHI [15] apresentou um estudo de caso na IBM, onde investigou a utilidade do TDD do ponto de qualidade de software e produtividade. Como resultado o estudo de caso apontarou que a qualidade foi maior nos projetos que utilizaram TDD. No estudo realizado já existia um processo de teste e o TDD foi incorporado ao processo existente. O trabalho apresentado neste artigo incorporou o TDD e testes de regressão em um processo de desenvolvimento sem nehuma etapa formal de testes.

Outro estudo semelhante foi realizado na cidade de Três de Maio, no Rio Grande do Sul, por STEFFENS [14]. Por se tratar de um município da mesma região de Frederico Westphalen o contexto e a cultura organizacional se tornam parecidos. O objetivo do trabalho era aplicar o TDD em uma empresa de pequeno porte. Entretanto, o TDD não obteve êxito, pois a empresa utiliza uma ferramenta de desenvolvimento de software baseada em conhecimento GeneXus [26]. A solução encontrada foi apenas utilizar teste de unidade. Nesse artigo além da proposta ser baseado no TDD, também foi utilizado teste de unidade automatizados.

LUFT [22], apresentou exemplos de aplicações em implementação de teste de software em empresas de pequeno, médio e grande porte, através de metodologias ágeis. O teste é uma das atividades de garantia de qualidade mais utilizada e de grande importância no desenvolvimento de um software. Através dele é possível encontrar falhas nos sistemas, antes que o mesmo seja entregue ao cliente. Esse trabalho apresenta um revisão das metodologias de teste utilizadas pelas empresas. Nosso trabalho focou apenas no TDD e técnicas simples de teste.

FERREIRA [23] abordou a utilização da metodologia TDD para desenvolvimento de software. Nesse podemos notar o uso do TDD de forma clara conforme o desenvolvimento das aplicações. O autor mostra que o processo do TDD pode aumentar a qualidade do software, diminuindo os custos. Em comparação no trabalho apresentado neste artigo, foi abordado tanto o TDD como as metologias de teste no contexto de uma empresa real. Foram utilizados os mesmos conceitos no entanto nesse trabalho sempre se levando as necessidades da empresa.

WILLIANS [24] apresentou a aplicação do TDD em 4 equipes de trabalho na indústria. O trabalho apresentou resultados e experiências de quatro equipes na indústria, onde foi desenvolvido quatro estudos de casos, aplicados em três equipes da Microsoft e uma equipe da IBM. O resultado desse estudo mostra que tanto na Microsoft quanto na IBM ocorreu um aumento entre 15% a 35% no tempo de desenvolvimento. Porém o aumento no tempo de desenvolvimento se justifica pelo aumento da confiabilidade dos produtos em 90%. Fica bem claro no trabalho que estas empresas possuem pessoas altamente qualificadas e com conhecimento pronto para operarem em várias metodologias diferentes não apenas no TDD. SANCHEZ [25] apresentou uma metodologia de desenvolvimento orientado a testes na IBM. Neste trabalho é possível visualizar o passo a passo de uma equipe que defende o uso do TDD há mais de cinco anos desenvolvendo aplicações e realizando testes manuais e automáticos ambos usando a metodologia TDD. O autor deixa explícito que com o uso do TDD são desenvolvidos produtos com alta qualidade. Observou-se nesses trabalhos que o contexto industrial influenciou de forma positiva a aplicação de metodologias como o TDD. Uma das razões é a estrutura das empresas e a facilidade de adequação a novas tecnologias e processos. Nesse sentido o trabalho proposto nesse artigo apresentou dificuldades justamente pela falta de pessoas especializadas na área e pelo processo de desenvolvimento tradicional e utilizado por muitos anos sem mudanças.

XI. CONCLUSÕES

Nesse artigo foi apresentado uma proposta de um processo de teste de software para uma empresa de pequeno porte. Utilizou-se como base a metodologia TDD. No entanto observamos que a empresa trabalha com muitos sistemas legados e precisava de outras técnicas de teste para os sistemas já existentes. Dessa forma foi utilizado teste de regressão para o sistema legado. Também foi demonstrado como automatizar os testes através da ferramenta DUnit.

Esse trabalho apresentou o TDD e sua importância e como ele poderia ser utilizado em uma empresa. Como principal conclusão, observou – se que apenas o TDD não é suficiente, necessita-se de outras técnicas de teste principalmente quando existe sistemas legados.

No desenvolvimento do projeto foram encontradas algumas dificuldades na realização como:

- Exemplificar a importância de testes de software no atual processo da empresa.
- Formalizar o processo do TDD.
- Possível mudança de cultura organizacional da empresa
- Tempo para se adaptar a mudanças.

Com a adequação à proposta do processo de testes padronizado, a empresa terá um ganho significativo: desde sua produtividade durante todo o novo processo de testes, além da união da equipe em função de um produto de maior qualidade, à padronização de processos internos e a otimização do produto.

Como trabalhos futuros pode se citar a aplicação da proposta na empresa por meio de um projeto piloto. Outros trabalhos futuros podem inclui investigação de outras técnicas de teste e a comparação com diferentes metodologias.

XII. REFERÊNCIA BIBLIOGRAFICAS

[1] MYERS, Glen. *The Art of Software Testing*. New York: Wiley, 1979.

[2] HETZEL, Bill. *The Complete Guide to Software Testing*. Ed. 2, John Wiley & Sons, 1988.

[3] SOMMERVILLE, I. *Engenharia de Software*, Addison Wesley, 8ª edição, 2007

[4] TOGNOLO, Andrei de Oliveira. *Teste automatizados com JUnit.* Revista Easy Java Magazine. São Paulo, Ed. 38, Online.

[5] DELAMARO, Márcio; MALDONADO, José; JINO, Mario. *Introdução ao Teste de Software.* Serie Campus. Campus. 2007. P. 1.

[6] ANICHE, Mauricio. *Test Driven Development – Teste e Design no Mundo Real.* Casa do Código. 2013. P.30.

[7] LOPES, Camilo. *Princípios de Test Driven Development (TDD).* iMasters. 2013. Disponível em: < http://imasters.com.br/artigo/24242/desenvolvimento/principios-de-test-driven-development-tdd/> Acessado em: 06/10/2014

[8] ANICHE, Mauricio. *Test Driven Development – Teste e Design no Mundo Real.* Casa do Código. 2013. P.28.

[9] PEZZÈ, Mauro, YOUNG, Michal. *Software Testing and Analysis - Process, Principles and Techniques*. Wiley 2007. P.211.

[10] PEZZÈ, Mauro, YOUNG, Michal. *Software Testing and Analysis - Process, Principles and Techniques.* Wiley 2007. P.161.

[11] ANICHE, Mauricio. *Test Driven Development – Teste e Design no Mundo Real.* Casa do Código. 2013. P.29.

[12] MORRIS, Chrismo. *DUnit: Xtreme Unit Testing for Delphi*. Github. Disponível em < http://sourceforge.net/projects/dunit/> Acessado em: 01/11/2014.

[13] CAETANO. Cristiano. *Melhores Práticas e Desafios na Automação de Testes* Disponível em: <http://www.qualister.com.br/blog/melhores-praticas-e-desafios-na-automacao-de-testes> Acessado em: 12/11/2014.

[14] STEFFENS. Cristiano. *Estudo de TDD e Aplicação de Testes Unitários Automatizados em Empresa de Desenvolvimento de Sistemas.* Sociedade Educacional Três de Maio. 2012.

[15] FRONZATHI. Alencar. *Avaliação de Eficácia do Teste Driven Development.* Disponível em: <http://www.devmedia.com/ensaios/Avalia%C3%A7%C3%A3o-Da-Efic%C3%A1cia-Do-Test-Driven-development/51934097.html> Acessado em: 07/10/2014.

[16] PRIMÃO, Aline. RIBEIRO, Patric. KREUTZ, Diego. *Estudo de Caso: Técnicas de Teste como parte do Ciclo de Desenvolvimento de Software.* Unipampa 2013.

[17] INTHURN, Cândida. *Qualidade & Teste de Software*. Visual Books. 2012.

[18] BARTIÉ, Alexandre. *Garantia de Qualidade de Software: adquirindo maturidade organizacional*. Rio de Janeiro: Editora Campus, 2002.

[19] INTHURN, Cândida. *Qualidade & Teste de Software*. Visual Books. 2012.

[20] CRESPO, Adalberto Nobiato. *Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo*. São Paulo, 2004.

[21] MACHADO, Alex. *A Implantação de Testes Automatizados de Aplicações Web num Ambiente Hostil.* Universidade Federal de Santa Catarina (UFSC), 2009.

[22] LUFT, Cristiane. *Teste de Software: Uma Necessidade das Empresas*. Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ), 2012.

[23] FERREIRA, Allan Rett. *Utilização da Metodologia TDD para o Desenvolvimento de Software*. Centro Universitário Eurípedes de Marilía (UNIVEM), 2011.

[24] WILLIANS, Laurie. MAXIMILLIEN, Michael. *Realizing Quality Improvement Through Test Driven Development: Results and Experiences of Four Industrial Teams.* Emp. Software Eng. 2008.

[25] SANCHEZ, Julio. WILLIANS, Laurie. MAXIMILLIEN, Michael. *On The Sustained Use of a Test-Driven Development Practice at IBM.* North Caroline State University. 2007.

[26] GENEXUS. Disponível em: <http://www.genexus.com/global/inicio?pt> Acessado em: 29/11/12.