

UNIVERSIDADE FEDERAL DO PAMPA
CAROLINE WASCHBURGER DOS SANTOS

**PROCESSO DE V&V APLICADO AO
DESENVOLVIMENTO DE SOFTWARE DO
NTIC**

ALEGRETE

2015

CAROLINE WASCHBURGER DOS SANTOS

**PROCESSO DE V&V APLICADO AO
DESENVOLVIMENTO DE SOFTWARE DO NTIC**

Projeto de Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Me. Sam Da Silva Devincenzi

ALEGRETE

2015

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

D312p Dos Santos, Caroline Waschburger
Processo de V&V Aplicado ao Desenvolvimento de Software do
NTIC / Caroline Waschburger Dos Santos.
88 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade
Federal do Pampa, ENGENHARIA DE SOFTWARE, 2015.
"Orientação: Sam da Silva Devincenzi".

1. Engenharia de Software. 2. Verificação e Validação. 3.
Testes. I. Título.

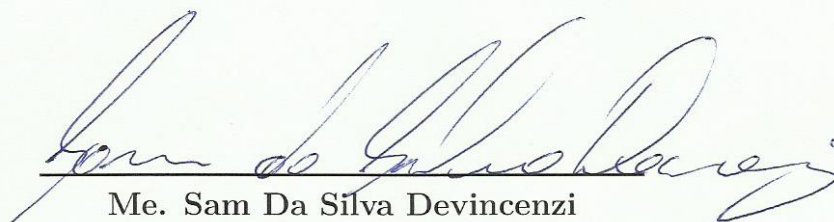
CAROLINE WASCHBURGER DOS SANTOS

PROCESSO DE V&V APLICADO AO DESENVOLVIMENTO DE SOFTWARE DO NTIC

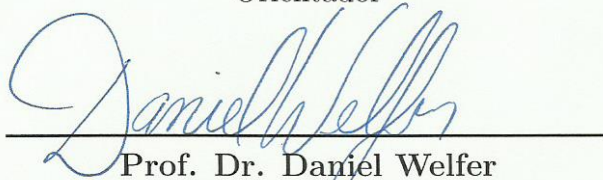
Projeto de Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Projeto de Trabalho de Conclusão de Curso defendido e aprovado em 22 de janeiro de 2015.

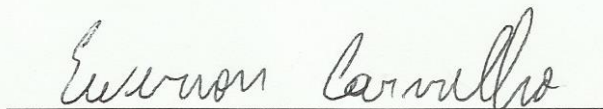
Banca examinadora:



Me. Sam Da Silva Devincenzi
Orientador



Prof. Dr. Daniel Welfer
Unipampa



Prof. Dr. Ewerson Luiz de Souza
Carvalho
Unipampa

*Este trabalho é dedicado a minha família,
fonte de inspiração e toda minha dedicação.*

Agradecimentos

Gostaria de agradecer primeiramente a minha mãe Elídes, que dedicou a vida dela a minha educação e me ensinou os valores que eu jamais esquecerei. Também agradeço ao meu pai Sílvio, que forneceu apoio emocional de forma incondicional durante esses anos. Aos meus segundos pais, Suzane e Jorge, muito obrigada por todos os momentos de uma vida e por me ajudarem a conquistar esse sonho.

À minha irmã de coração Gabriele e a toda minha família, que durante os últimos meses escutou pacientemente minhas reclamações. Em especial minha avó, que tanto insistiu em minha graduação.

Agradeço ao meu namorado Daniel que demonstrou paciência e ofereceu constantemente um ombro amigo.

Aos meus colegas e amigos, gostaria de dizer que o caminho que trilhamos foi inesquecível. Miguel, obrigada pela paciência ao ensinar programação. Matheus, companheiro de RP e piadista, agradeço por toda sua ajuda e risadas. Rafael, sem palavras para demonstrar toda minha gratidão pela sua ajuda. Greicy, quero agradecer por todas as horas, trabalhos, risadas e companheirismo durante esses 4 anos. Muito obrigada amigos!

A todo pessoal da CODEV, pela hospitalidade, ajuda e oportunidade em aprender com grandes profissionais.

Por fim, gostaria de agradecer ao meus professores de Engenharia de Software pela dedicação ao transmitir o conhecimento adquirido, em especial ao meu orientador Sam que foi imprescindível para a realização desse trabalho.

*O período de maior ganho em conhecimento e experiência
é o período mais difícil da vida de alguém.*

Dalai Lama

Resumo

A crescente execução de tarefas, simples ou complexas, a partir de um software demanda que o mesmo possua algumas características, como uma boa usabilidade, confiabilidade e eficiência. Com a compreensão dessas necessidades é possível afirmar que o processo de desenvolvimento de um software exige muito mais do que analisar as funcionalidades requisitadas e implementá-las, exige um conjunto de atividades robustas, que podem ser aplicadas de muitas formas, onde o resultado final esperado deve ser sempre o mesmo: um software com qualidade. A partir desse contexto, o presente trabalho se propõe a elaborar e aplicar um processo de verificação e validação para o NTIC- CODEV. A elaboração do processo foi feita a partir de uma pesquisa literária quanto as técnicas de V&V, compreensão de trabalhos relacionados e identificação de lacunas no desenvolvimento da CODEV. Com o processo modelado, selecionamos as funcionalidades disponíveis para aplicação e as atividades do processo foram executadas sequencialmente. Os dados gerados na aplicação foram monitorados e analisados. Os resultados foram quantificados seguindo o indicador de retrabalho ou seja, os defeitos encontrados foram contabilizados assim como o tempo de correção exigido. Os cálculos realizados indicam o custo da produção com e sem retrabalho.

Palavras-chave: Engenharia de Software. Verificação e Validação. Testes.

Abstract

The growing executing tasks, simple or complex, from software demand that it has some characteristics such as good usability, reliability and efficiency. With the understanding of these requirements is possible to say that the process of developing a software requires much more than the required functionality to analyze and implement them, requires a robust set of activities that can be applied in many ways, where the final result expected should always be the same: software quality. From this context, the present work aims to develop and apply a process of verification and validation. The process was made from a literary research about the techniques of V&V, understanding of related jobs and identification of gaps in development of CODEV. With the modeled process, select the features available to application and process activities were executed sequentially. The data generated in the application were monitored and analyzed. The results were quantified following rework indicator, i.e. the defects found were recorded as well as the time of correction required. The calculations indicate the cost of production with and without rework.

Key-words: Software Engineering. Verification and Validation. Tests.

Lista de ilustrações

| | |
|--|----|
| Figura 1 – Camadas da Engenharia de Software. | 18 |
| Figura 2 – Processo de Inspeção de Software. | 22 |
| Figura 3 – Processo de Desenvolvimento Antigo da CODEV | 30 |
| Figura 4 – Novo Processo de Desenvolvimento da CODEV | 31 |
| Figura 5 – Atividade 1 - Inspeccionar Requisitos. | 32 |
| Figura 6 – Atividade 2 - Criar Casos de Teste. | 33 |
| Figura 7 – Atividade 3 - Inspeccionar Modelos. | 34 |
| Figura 8 – Atividade 4- Testes Unitários. | 34 |
| Figura 9 – Atividade 5 - Inspeccionar a Estrutura do Código | 35 |
| Figura 10 – Atividade 6 - Inspeccionar Implementação do Projeto no Código. | 36 |
| Figura 11 – Atividade 7- Testes de Aceitação. | 37 |
| Figura 12 – Requisitos do pacote solicitados pela CAU. | 39 |
| Figura 13 – Caso de Uso U002- Visualizar Plano de Ensino (Aluno). | 39 |
| Figura 14 – Informações, Objetivos e Pré-condições dos casos de teste visualizar plano de ensino. | 40 |
| Figura 15 – Cenário, Entradas e Resultados Esperados dos casos de teste Visualizar Plano de Ensino | 40 |
| Figura 16 – Checklist para Inspeção Guiada de Código Referente a Visualizar Plano de Ensino. | 41 |
| Figura 17 – Visualizar Plano de Ensino- Resultado Obtido com o teste de aceitação. | 42 |
| Figura 18 – Visualizar Plano de Ensino- Resultado Obtido com o reexecução dos testes. | 43 |
| Figura 19 – Caso de Uso U002- Anexar Arquivos. | 44 |
| Figura 20 – Informações quanto aos Casos de teste Manter PSA. | 44 |
| Figura 21 – Caso de Teste Manter PSA- Continuação. | 45 |
| Figura 22 – Caso de teste Manter PSA- Resultados Esperados. | 45 |
| Figura 23 – Diagrama de Sequência Anexar Arquivos | 46 |
| Figura 24 – Checklist Utilizado no Pacote 1387. | 47 |
| Figura 25 – Resultado da execução do Caso de teste Anexar Arquivos. | 47 |
| Figura 26 – Caso de Uso 004- Inserir Observações no Plano de Trabalho(chefia imediata). | 48 |
| Figura 27 – Informações do Caso de Teste Inserir Observações no Plano de Trabalho. | 49 |
| Figura 28 – Caso de Teste Inserir Observações no Plano de Trabalho - Continuação. | 49 |
| Figura 29 – Casos de teste - Entrada e Resultados Esperados. | 50 |
| Figura 30 – Diagrama de classes modelado para o plano de trabalho. | 50 |
| Figura 31 – Checklist utilizado no código do módulo Plano de Trabalho. | 51 |

| | |
|--|----|
| Figura 32 – Testes de Aceitação Referente ao Caso de Uso Inserir Observações no Plano de Trabalho. | 52 |
| Figura 33 – PTP , EVT e RU - Total de Defeitos Encontrados | 54 |
| Figura 34 – PTP , EVT e RU - Custo Estimado de Desenvolvimento. | 54 |
| Figura 35 – PTP , EVT e RU - Distribuição do Custo Estimado de Desenvolvimento . | 55 |
| Figura 36 – PTA, PSA e PTR - Distribuição do Custo Estimado de Desenvolvimento | 56 |
| Figura 37 – PTA, PSA e PTR - Distribuição do Custo Estimado de Desenvolvimento | 56 |
| Figura 38 – PTA, PSA e PTR- Distribuição do Custo Estimado de Desenvolvimento | 57 |
| Figura 39 – Custos de Desenvolvimento Considerando Correção a 14,6%. | 57 |

Lista de tabelas

| | |
|--|----|
| Tabela 1 – Definições da literatura para VVT | 19 |
|--|----|

Sumário

| | | |
|----------|--|-----------|
| 1 | INTRODUÇÃO | 15 |
| 1.1 | Justificativa | 16 |
| 1.2 | Objetivos | 16 |
| 1.3 | Metodologia | 17 |
| 1.4 | Organização do Documento | 17 |
| 2 | FUNDAMENTAÇÃO TEÓRICA | 18 |
| 2.1 | Engenharia de Software | 18 |
| 2.2 | Verificação, Validação e Testes | 19 |
| 2.2.1 | V&V Estática | 20 |
| 2.2.2 | V&V Dinâmica | 22 |
| 2.3 | Conclusão do Capítulo | 24 |
| 3 | TRABALHOS RELACIONADOS | 25 |
| 3.1 | Trabalhos Relacionados a ERP | 25 |
| 3.2 | Trabalhos Relacionados a V&V de Software | 26 |
| 3.3 | Trabalhos Relacionados a Testes de Software | 26 |
| 3.4 | Conclusão do Capítulo | 27 |
| 4 | PROCESSO DE V&V PARA O NTIC | 29 |
| 4.1 | Processo Antigo | 29 |
| 4.2 | Novo Processo Proposto | 29 |
| 4.2.1 | Atividade 1 - Inspeccionar requisitos | 32 |
| 4.2.2 | Atividade 2 - Criar Casos de Teste | 32 |
| 4.2.3 | Atividade 3 - Inspeccionar Modelos | 33 |
| 4.2.4 | Atividade 4 - Teste Unitário | 34 |
| 4.2.5 | Atividade 5- Inspeccionar Código | 35 |
| 4.2.6 | Atividade 6 - Inspeccionar Código | 36 |
| 4.2.7 | Atividade 7- Testes de Aceitação | 36 |
| 4.3 | Conclusão do Capítulo | 37 |
| 5 | ESTUDO DE CASOS | 38 |
| 5.1 | #1529 Pacote Para o Módulo PTA (Portal do Aluno) | 38 |
| 5.1.1 | Inspeccionar os requisitos | 38 |
| 5.1.2 | Criar Casos de Testes | 40 |
| 5.1.3 | Inspeccionar Modelos | 41 |

| | | |
|------------|---|-----------|
| 5.1.4 | Inspeccionar código: estrutura | 41 |
| 5.1.5 | Testes de Aceitação | 42 |
| 5.2 | #1635 Pacote Para o Módulo PSA (Processo Seletivo Acadêmico) | 43 |
| 5.2.1 | Inspeccionar os requisitos | 43 |
| 5.2.2 | Criar Casos de Testes | 44 |
| 5.2.3 | Inspeccionar Modelos | 45 |
| 5.2.4 | Inspeccionar código: estrutura | 46 |
| 5.2.5 | Testes de Aceitação | 46 |
| 5.3 | #1387 Pacote para o Módulo PTR (Plano de Trabalho) | 48 |
| 5.3.1 | Inspeccionar os requisitos | 48 |
| 5.3.2 | Criar Casos de Testes | 49 |
| 5.3.3 | Inspeccionar Modelos | 50 |
| 5.3.4 | Inspeccionar código: estrutura | 51 |
| 5.3.5 | Testes de Aceitação | 52 |
| 5.4 | Conclusão do Capítulo | 52 |
| | | |
| 6 | RESULTADOS | 53 |
| | | |
| 7 | CONCLUSÃO | 59 |
| 7.1 | Trabalhos Futuros | 59 |
| | | |
| | ANEXOS | 60 |
| | | |
| | ANEXO A – CHECKLIST PARA INSPEÇÃO DE CÓDIGO ELABORADO PELA CODEV | 61 |
| | | |
| | APÊNDICES | 64 |
| | | |
| | APÊNDICE A – TEMPLATE PARA CASOS DE TESTE | 65 |
| | | |
| | APÊNDICE B – 1529 ATIVIDADE 1 - SAÍDA DA INSPEÇÃO DE REQUISITOS | 67 |
| | | |
| | APÊNDICE C – 1529 ATIVIDADE 2 E 7 - SAÍDA DA CRIAÇÃO DOS CASOS DE TESTE E TESTES DE ACEITAÇÃO EXECUTADOS | 69 |
| | | |
| | APÊNDICE D – 1529 ATIVIDADE 5 - SAÍDA DA INSPEÇÃO DE CÓDIGO | 71 |

| | |
|---|----|
| APÊNDICE E – 1635 ATIVIDADE 1 - SAÍDA DA INSPEÇÃO DE REQUISITOS | 74 |
| APÊNDICE F – 1635 ATIVIDADE 2 E 7 - CRIAÇÃO DOS CASOS DE TESTE E TESTES DE ACEITAÇÃO EXECUTADOS | 77 |
| APÊNDICE G – 1635 ATIVIDADE 5 - INSPEÇÃO DE CÓDIGO | 79 |
| APÊNDICE H – 1387 ATIVIDADE 1 - SAÍDA DA INSPEÇÃO DE REQUISITOS | 82 |
| APÊNDICE I – 1387 ATIVIDADE 2 E 7 - CRIAÇÃO DOS CASOS DE TESTE E TESTES DE ACEITAÇÃO EXECUTADOS | 85 |
| Referências | 87 |

1 Introdução

A utilização dos sistemas computacionais trouxe constante progresso e com ele possibilidades e facilidades para o dia a dia, não só de empresas mas também de todas as pessoas. Essas facilidades proporcionaram tantos benefícios à sociedade moderna que nos tornamos dependentes desses sistemas e estamos sempre procurando aperfeiçoá-los.

Um dos componentes fundamentais de um sistema computacional, e objeto de estudo da Engenharia de Software, é o software. De acordo com (SOMMERVILLE, 2007, p. 4) "Software não é apenas o programa, mas também todos os dados de documentação e configurações associados, necessários para que o programa opere corretamente", ou seja, a definição de software vai além do que um subsistema de um sistema computacional, são programas, processos, normas e toda a documentação correspondente.

A crescente execução de tarefas, simples ou complexas, a partir de um software demanda que o mesmo possua algumas características, como uma boa usabilidade, confiabilidade e eficiência. Com a compreensão dessas necessidades é possível afirmar que o processo de desenvolvimento de um software exige muito mais do que analisar as funcionalidades requisitadas e implementá-las, exige um conjunto de atividades robustas, que podem ser aplicadas de muitas formas, onde o resultado final esperado deve ser sempre o mesmo: um software com qualidade.

Para (BOURQUE; FAIRLEY, 2004) a qualidade do software é obtida através da conformidade com todos os requisitos, independentemente de que característica é especificada ou como os requisitos são agrupados ou nomeados.

Além da utilização de um processo bem definido para desenvolvimento de software, as atividades de Verificação e Validação (V&V) tem por objetivo garantir que o sistema desenvolvido está de acordo com o que foi requisitado, conferindo confiabilidade ao sistema (SOMMERVILLE, 2007). A partir disso é possível compreender que essas atividades estão relacionadas/consideradas com a garantia de qualidade de software, sendo atividades de fundamental importância para tal.

O teste de software é considerado a técnica principal dentro do processo de V&V, pois testar tem a finalidade de descobrir defeitos no sistema e demonstrar que o software está de acordo com seus requisitos (SOMMERVILLE, 2007). Os diversos tipos de testes existentes podem ser aplicados para diferentes finalidades, quando utilizados em conjunto é possível que sejam descritos como um processo. Conforme (MYERS, 2004, p. 1-2) "O teste de software é um processo, ou uma série de processos, desenvolvido para garantir que o código de computador faz o que foi projetado para fazer e que não faz nada não intencional. O software deve ser previsível e consistente, não oferecendo surpresas ao

usuário."

A Universidade Federal do Pampa (UNIPAMPA) dispõe de um núcleo para gestão de informação e desenvolvimento de software. O Núcleo de Tecnologia e Comunicação (NTIC) é subdividido em coordenadorias. Dentre elas, a Coordenadoria de Desenvolvimento (CODEV) que é responsável pelo desenvolvimento de um sistema de gestão integrada, também conhecido como *Enterprise Resource Planning* (ERP), dentro da universidade.

Um sistema ERP é basicamente um software de gestão que integra vários módulos de uma empresa, dispensando a necessidade de se ter um sistema isolado para cada departamento. Pode-se afirmar que o seu desenvolvimento precisa de um processo eficaz para garantir qualidade e confiabilidade a esses sistemas. Para tanto, esse trabalho tem por finalidade desenvolver um processo de V&V contribuindo assim, para atividade de desenvolvimento de software no CODEV-NTIC.

1.1 Justificativa

Retrabalho em desenvolvimento de software implica em reexecutar alguma atividade a qual não resultou no correto produto em sua anterior execução. O Retrabalho é uma das principais causas de problemas de cronograma em projetos de software, fazendo com que em muitos casos prazos não sejam cumpridos e custos sejam aumentados. Má especificação de requisitos e defeitos encontrados ao longo do desenvolvimento estão entre as causas para o retrabalho.

Seja por má especificação vinda da Coordenadoria de Apoio ao Usuário (CAU) ou por defeitos decorrentes do desenvolvimento com pouca presença de atividades para garantia de qualidade, a CODEV acaba vivenciando o retrabalho. Sendo assim inserir um processo de V&V na metodologia de desenvolvimento da CODEV, pode diminuir a incidência de retrabalhos.

1.2 Objetivos

A busca por atividades que possam auxiliar a garantia de qualidade no desenvolvimento de software não é somente algo adicional para esse processo, e sim essencial. Nesse contexto, o objetivo geral desse trabalho é determinar um processo de verificação e validação para garantia de qualidade dos software desenvolvidos pelo NTIC. Para isso, os seguintes objetivos específicos também deverão ser alcançados:

- Realizar um estudo das técnicas de V&V;
- Realizar um estudo dos tipos de teste;

- Realizar estudo do processo de VV&T da CODEV;
- Propor um novo processo de V&V para CODEV;
- Propor pontos de avaliação do novo processo;
- Validar o processo de V&V.

1.3 Metodologia

Para que os objetivos do trabalho proposto sejam alcançados, o desenvolvimento contou com pesquisa bibliográfica, onde foi realizado um estudo das práticas de V&V existentes, e do aspecto de campo, estudo de caso, onde compreendemos o modelo de processo utilizado no NTIC. Posteriormente, foi elaborada uma proposta de processo de V&V para o NTIC e definidos os indicadores para medir a eficiência deste processo.

Com a proposta concluída, realizamos a execução deste processo em alguns projetos do NTIC e dados foram coletados, servindo de análise juntamente com os indicadores preestabelecidos.

1.4 Organização do Documento

No Capítulo 1 foi abordada uma introdução ao tema alvo deste trabalho. Também foram descritos os objetivos que impulsionaram essa pesquisa e a metodologia em que o trabalho foi construído.

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta um revisão da literatura utilizada para apoiar a construção do processo. O Capítulo 3 descreve alguns trabalhos relacionados ao tema deste trabalho. O Capítulo 4 descreve o processo antigo e novo que foi elaborado para a CODEV. O Capítulo 5 descreve o relato da execução do processo em alguns projetos da CODEV. O resultados obtidos com a execução do processo na CODEV são descritos no Capítulo 6. O Capítulo 7 apresenta algumas considerações finais e trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo apresenta-se a fundamentação teórica contendo definições que servem de base para o desenvolvimento deste trabalho. A seção 2.1 aborda uma contextualização sobre Engenharia de Software. A seção 2.2 descreve conceitos relacionados as atividades de Verificação, Validação e Testes de software. A subseção 2.2.1 aborda as técnicas estáticas e a subseção 2.2.2 aborda as técnicas dinâmicas para V&V.

2.1 Engenharia de Software

O termo "**Engenharia de Software**" surgiu após a conferência NATO¹ em 1968, descreveu-se a Engenharia de Software para solucionar aumento da demanda de desenvolvimento de software e os problemas que isso acarretou. Um dos motivos para o surgimento foi o fato do software se tratar de algo abstrato e seu desenvolvimento não adaptar-se a nenhuma das engenharias (WAZLAWICK, 2013).

Com o intuito de suprir as necessidades existentes no ciclo de desenvolvimento e obter um produto final pensando em qualidade e produtividade, a Engenharia de Software engloba processos, conjunto de métodos ou práticas e diversas ferramentas.

Para (PRESSMAN, 2011) a Engenharia de Software está relacionada a uma tecnologia em camadas, onde o foco está na qualidade. A Figura 1 demonstra visualmente essa organização em camadas, com a qualidade sendo representada pela base onde estão apoiados processos, métodos e ferramentas.

Figura 1 – Camadas da Engenharia de Software.



Fonte: Pressman (2011, p. 39)

O processo de desenvolvimento de um software está associado as etapas para colaborar com construção de um sistema. Essas etapas devem conter atividades de análise,

¹ NATO Software Engineering Conference realizada em outubro de 1968 na cidade de Garmisch, Alemanha

projeto, implementação e testes. Há inúmeros processos na literatura que abordam outras atividades e formas diferentes de implementá-las, porém cada empresa de desenvolvimento deve escolher o processo mais adequado a seu propósito e manter o foco na qualidade, tanto do produto quanto do processo em si.

Com base no que foi mencionado, (PRESSMAN, 2006) afirma que: "Os engenheiros de software buscam qualidade (e desenvolvem atividades de garantia de qualidade e de controle de qualidade) aplicando métodos e medidas técnicas sólidas, conduzindo revisões técnicas formais e efetuando teste de software bem planejado."

Para seguir no escopo deste trabalho, iremos abordar as atividades de V&V que são parte do processo de desenvolvimento e servem como garantia de qualidade.

2.2 Verificação, Validação e Testes

A literatura dispõe de diversas definições envolvendo atividades para Verificação, Validação e Testes. Essas definições, em um contexto geral, são associadas a detecção de erros, garantia de qualidade e conformidade dos requisitos. A Tabela 1 apresenta alguns conceitos de VV&T.

Tabela 1 – Definições da literatura para VV&T.

| Autor | Definição |
|-------------------------------------|---|
| Ian Sommerville (SOMMERVILLE, 2007) | O papel da <i>verificação</i> envolve verificar se o software está de acordo com suas especificação. Você deve verificar se ele atende aos requisitos funcionais e não funcionais. A finalidade da <i>validação</i> é assegurar que o sistema de software atenda às expectativas do cliente. |
| Roger S. Pressman (PRESSMAN, 2006) | <i>Verificação</i> se refere ao conjunto de atividades que garante que o software implementa corretamente uma função específica. <i>Validação</i> se refere a um conjunto de atividades diferentes que garante que o Software construído corresponde aos requisitos do cliente. |
| Raul S. Wazlawick (WAZLAWICK, 2013) | <i>Verificação</i> : consiste em analisar o software para ver se ele está sendo construído de acordo com o que foi especificado. <i>Validação</i> : consiste em analisar o software construído para ver se ele atende às verdadeiras necessidades dos interessados. |
| Swebok (BOURQUE; FAIRLEY, 2004) | [...] técnica para avaliação da qualidade de um produto, e consequentemente, sua melhoria através da identificação de defeitos e problemas. |

Fonte: Elaborado pelo autor.

Conforme as definições acima, as atividades de VV&T estão inseridas dentro da Garantia de Qualidade de Software (SQA), sendo assim é importante destacar que essas atividades podem ocorrer a partir do momento da concepção do sistema, e não somente em uma etapa final para encontrar defeitos.

O processo VV&T define que cada produto ou artefato elaborado durante o ciclo de desenvolvimento do software seja avaliado, verificado, validado e testado em cada etapa antes de seguir para a próxima etapa, assegurando que os mesmos estejam completos e corretos, seguindo as exigências de verificação e validação.

A garantia da qualidade é uma forma de fazer com que o produto e o processo contemplem os aspectos entendidos como qualidade. Entretanto, conceituar Qualidade de Software é um pouco mais complicado do que se espera, pois difere daquela definição encontrada no dicionário que esta atrelada a características das "coisas" que a tornam diferente das outras. Uma definição considerada abrangente é dada por (BARTIE, 2002, p. 16): "Qualidade de software é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos".

As atividades de V&V são comumente divididas em estáticas e dinâmicas. A diferenciação dessas duas denominações pode ser feita de forma simples: as **estáticas** são atividades que não necessitam a execução do software em desenvolvimento para serem efetuadas e as **dinâmicas** são as que se baseiam na execução.

2.2.1 V&V Estática

A verificação e a validação incluem muitas atividades relacionadas a garantia de qualidade (SQA), como : revisões, auditorias de qualidade e configuração, monitoramento de desempenho, simulação, estudo de viabilidade, revisão de documentos, entre outras (PRESSMAN, 2011). Dentre essas atividades podemos destacar as revisões técnicas.

As revisões são atividades de análise que podem ser aplicadas em qualquer artefato de software, desde documentos de especificação de requisitos até os códigos fonte. (PRESSMAN, 2011) compara as revisões como filtros que servem para revelar erros e defeitos.

A execução de uma revisão normalmente envolve um grupo de pessoas que examina o software e a documentação associada à procura de possíveis problemas e não conformidade com padrões (SOMMERVILLE, 2011), porém também é possível efetuar-las de modo informal, como uma conversa onde sejam discutidos problemas técnicos.

Há mais de uma forma ou objetivo para realizar uma revisão, portanto elas são divididas de acordo com suas características que diferem. As mais conhecidas são revisões informais, inspeções e walkthroughs. A seguir, discute-se cada uma delas.

Inspeções de Software

A inspeção de software foi originalmente definida por (FAGAN, 1986) como "um tipo específico de revisão que tem a função de detectar defeitos antes que a fase de teste seja iniciada, contribuindo para a melhoria da qualidade geral do software."

Outra definição a ser considerada para entender a metodologia dessa técnica é a descrita pela (IEEE, 2008):

"A inspeção, mais formal que a revisão técnica, tem o objetivo principal de identificação e remoção de defeitos. É obrigatória a geração de uma lista de defeitos com a classificação padronizada, requerendo-se a ação dos autores para a remoção desses defeitos. No Praxis padrão, são aplicadas aos artefatos de desenho, implementação e testes, focalizando a correção desses em relação aos respectivos padrões e especificações, enquanto as revisões técnicas têm maior enfoque na qualidade da documentação."(IEEE, 2008, p.)

O universo da inspeção é composto por papéis, processos e técnicas de leitura, isto é, por quem são os inspetores, como organizam seu trabalho e sincronizam suas atividades, e como examinam os artefatos a serem inspecionados (PEZZÈ; YOUNG,). Para que essa atividade funcione da maneira correta e retorne os resultados esperados é importante que todos os envolvidos estejam habituados com o processo de inspeção.

A proposta original para os papéis da equipe que realiza a inspeção é composta por autor, leitor, testador e moderador (FAGAN, 1986). Porém, a concepção que se tem hoje é de que a equipe deve ser composta com pelo menos 4 pessoas, onde a mesma pessoa pode desempenhar mais de um papel, e esses papéis podem exercer as respectivas funções:

- **Autor:** É o responsável por ter desenvolvido o artefato e também responsável pelo retrabalho gerado pela descoberta de defeitos, ou seja, corrigir os erros encontrados.
- **Inspetor:** Tem a função de encontrar erros ou inconsistências nos artefatos separados para inspeção. Todos os participantes podem desempenhar o papel de inspetor.
- **Relator:** Desempenha a função de registrar os resultados da reunião de inspeção.
- **Moderador:** Deve gerenciar e facilitar a inspeção. Também efetua o planejamento de todas as atividades de inspeção que fazem parte do projeto.

O processo de inspeção é simplificado pela Figura 2. De acordo com (SOMMERVILLE, 2007), o moderador deve efetuar o planejamento da inspeção, onde deve selecionar uma equipe, sala de reunião e garantir que o artefato a ser inspecionado esteja completo. Esse artefato é explicado a equipe de inspetores durante a etapa de visão geral, essa atividade é de responsabilidade do autor. Após isso há um período para a preparação individual, com o objetivo de que cada membro estude o artefato e procure por defeitos

Figura 2 – Processo de Inspeção de Software.



Fonte: (SOMMERVILLE, 2007)

no mesmo. A partir dos defeitos destacados o autor do artefato deve efetuar as correções necessárias. Posteriormente, o moderador deve analisar essas correções e decidir se uma nova inspeção deve ser feita ou se o artefato pode ser liberado.

Com a atividade de inspeção, há a detecção de defeitos nas fases iniciais do processo de desenvolvimento de software, facilitando a correção destes defeitos com menor esforço e custo. Desta forma, de acordo com (JONES, 2008), o esforço com retrabalho é reduzido em média para 10% a 20% do esforço total de desenvolvimento. Esta redução no retrabalho pode acrescentar melhorias consideráveis com relação a qualidade do software e nos custos do mesmo.

Revisão técnica

A Revisão Técnica tem a finalidade de avaliar artefatos pré-definidos para verificar se os mesmos estão de acordo com padrões e especificações. Num contexto de mudanças nesses artefatos é necessário verificar se foram efetuadas com sucesso (IEEE, 2008).

Walkthrough

O Walkthrough é um processo menos rigoroso se comparado à Inspeção. O autor deve apresentar o material seguindo uma ordem lógica, onde não há limite de tempo e a medida que vai sendo apresentado, os participantes devem efetuar as verificações. Portanto, esse tipo de reunião não necessita muita preparação antecipada e acaba sendo considerada de eficácia moderada para detecção de defeitos.

2.2.2 V&V Dinâmica

O teste de software é uma atividade dinâmica com o intuito de executar o programa com entradas específicas e verificar se seu comportamento está de acordo com o esperado. Ele é utilizado também para demonstrar a confiabilidade do software.

O processo básico de testes deve envolver uma mistura de testes manuais e automatizados. De acordo com (SOMMERVILLE, 2011) os testes automatizados são mais rápidos que os manuais, pois o fluxo segue a codificação dos testes em um programa e a execução sempre que necessário. Já no teste manual, o testador deve executar o programa desenvolvido com os dados de teste preparados e comparar os mesmos com o resultado

esperado.

A viabilidade para selecionar as técnicas devem estar de acordo com custos, prazos e recursos disponíveis no desenvolvimento. Cada projeto deve elaborar, avaliar e refinar um conjunto de técnicas que é adequado ao propósito do domínio do sistema.

As diversas técnicas conhecidas para exercer as atividades de testes podem ser agrupadas pelas seguintes questões: quando testar, o que testar e como testar.

Quando Testar

Está relacionado com a fase do desenvolvimento de software em que o teste será realizado.

- **Teste de Unidade:** O teste de unidade, ou teste de componentes, consiste em testar a menor unidade do software. Comumente esse tipo de teste é executado em pequenos trechos de código.
- **Teste de Integração:** O objetivo desse teste é verificar se os componentes funcionam quando estão em conjunto.
- **Teste de Sistema:** O teste de sistema avalia o software de forma integrada, como um usuário final.
- **Teste de Aceitação:** O teste de aceitação é aplicado por usuários ou a pela equipe de testes para verificar algumas funcionalidades e definir se as mesmas podem ser consideradas aceitas.
- **Teste de Regressão:** Deve ser executado, a cada nova versão do software, todos os testes que já foram realizados nas versões de teste anteriores do sistema.

O que testar

Estão agrupadas as técnicas que correspondem ao tipo de teste que será executado.

- **Teste de Funcionalidade:** É utilizado para testar as funções do sistema, garantindo que estão de acordo com o especificado nos requisitos funcionais.
- **Teste de Interface:** Verifica se a navegabilidade e os objetivos da tela funcionam como esperado.
- **Teste de Desempenho:** Esse tipo de teste permite por a capacidade de resposta de um sistema em determinados cenários e configurações.
- **Teste de Carga:** Analisa o funcionamento da aplicação exercendo sobre ela uma quantidade grande de usuários ao mesmo tempo.

- **Teste de Usabilidade:** O teste de usabilidade preocupa-se em avaliar a situação do sistema com relação a questões como facilidade no uso e outras associadas a interação humano computador.
- **Teste de Volume:** Tem a função de testar a quantidade de dados envolvidos (pouca, normal, grande, etc).
- **Teste de Segurança:** Serve para testar a segurança da aplicação ao utilizar diversos papéis, perfis, permissões, para executar o sistema.

Como testar

Como testar agrupa as técnicas que podem ser utilizadas ao testar, preocupando-se com as saídas encontradas ou com a estrutura interna do código.

- **Teste Funcional Caixa-Preta:** O teste de caixa-preta está relacionado com as entradas e saídas desejadas. A preocupação então não é com o código gerado e sim com a saída esperada e obtida.
- **Teste estrutural Caixa-Branca:** O teste caixa-branca é o oposto do caixa-preta, pois está diretamente preocupado com o código fonte e em conhecer a estrutura interna do sistema.

2.3 Conclusão do Capítulo

O Capítulo 2, em suas subseções, abordou os temas relevantes a Engenharia de Software, como a Verificação e Validação. Com a conclusão desse capítulo foi possível conhecer as técnicas mais utilizadas para garantia de qualidade e em que momento elas podem ser utilizadas dentro de um processo de desenvolvimento.

O Capítulo 3, na sequência, contém os trabalhos relacionados, que foram selecionados para compor a pesquisa.

3 Trabalhos Relacionados

Neste capítulo são apresentados alguns trabalhos relacionados que justificam o assunto deste trabalho e estão divididos em três áreas: o desenvolvimento de sistemas ERP, V&V de Software e Testes de Software.

O desenvolvimento de sistemas ERP, uma das áreas de pesquisa, foi selecionado devido a necessidade de compreendermos o processo de desenvolvimento desses sistemas. Tendo em vista que são sistemas que integram muitos módulos, pesquisar quanto aos tipos de testes mais indicados e compreender o impacto da qualidade no processo desses sistemas é essencial.

Compreender as aplicações da área de VV&T, utilizada como base para construção do processo, trouxe a possibilidade de verificar como as atividades são planejadas para se adaptar aos projetos e em quais momentos elas são mais utilizadas dentro do desenvolvimento.

3.1 Trabalhos Relacionados a ERP

(IFINEDO; NAHAR, 2006) realizou um estudo nomeado "Quality, Impact and Success of ERP Systems: A Study Involving Some Firms in the Nordic-Baltic Region", para explorar a qualidade, impacto e sucesso de sistemas de ERP, por amostragem, contendo o ponto de vista de duas empresas privadas da Finlândia e Estônia. A metodologia da pesquisa foi desenvolvida em três etapas incluindo tanto pesquisa qualitativa e quantitativa quanto abordagens para validade das conclusões. Na fase inicial, um questionário foi desenvolvido e enviado as empresas que se mostraram favoráveis em participar, posteriormente foram realizadas entrevistas com as empresas selecionadas. Os resultados obtidos apresentaram diferentes pontos de vista sobre a qualidade e impactos de ERP, o que pode ser ligado ao fato de como o ERP é adotado pelas empresas. Ainda, os resultados mostraram que a maioria das empresas valorizam a qualidade da informação das ERP e acreditam que o seu impacto organizacional é menor.

(KENNER; 2010) em sua monografia "Avaliação Da Qualidade De Software ERP De Acordo Com A Norma ISO/IEC 9126", onde o objetivo é avaliar a qualidade de produtos de software ERP tendo como base a norma ISO/IEC 9126. A metodologia de aplicação iniciou selecionado o modelo MEDE-PROS. Em seguida, o autor avaliou o produto de software ERP Compiere, que foi selecionado, de acordo com critérios pré-estabelecidos. Para finalizar o autor apresenta um plano de melhoria da qualidade do produto de software ERP Compiere. Os resultados do estudo de caso comprovaram a eficiência das seis ca-

racterísticas avaliadas do Compiere, apesar de terem sido identificados alguns problemas de qualidade. Entretanto o autor justifica que com algumas customizações e correções, o sistema ERP se mostra totalmente eficiente e pode ser a solução buscada para muitas empresas, em especial pequenas e médias empresas.

3.2 Trabalhos Relacionados a V&V de Software

(DESHMUKH; KAUSHIK, 2013) em seu artigo "A Overview of Software Verification and Validation and Selection Process"relata técnicas de verificação, validação e testes que podem ser utilizadas em todo o processo de desenvolvimento do software, de forma isolada ou a partir de uma combinação dessas técnicas. A definição das técnicas a serem utilizadas deve estar de acordo com fatores de importância e complexidade do sistema. Segundo o autor, o uso com sucesso dessas técnicas na indústria de desenvolvimento de software irá validar o resultado da pesquisa. Entretanto, o autor não aborda claramente quais possíveis combinações entre as técnicas são feitas na tentativa de reforçar a garantia da qualidade.

(SINDE; HO, 2008) em seu artigo "Independent Verification and Validation"propõe um órgão independente de verificação e validação, cujo papel é fornecer garantia de que um sistema atende aos requisitos e objetivos pré-definidos. A metodologia consiste em definir um projeto de ciclo de vida com as atividades de Verificação e Validação. O autor defende que quando elaborado da forma correta e incorporado ao ciclo de vida do desenvolvimento de sistemas é possível reduzir riscos, custos globais e garantir um bom desempenho do projeto.

(FERREIRA; MOITA, 2010) propõe em seu artigo "Inspection technique for the validation of software development processes", um método de validação para processos de desenvolvimento de software, baseado nos conceitos de Verificação e Validação. O método proposto faz uso da técnica de inspeção de software para validar esses processos. Porém a técnica de inspeção foi adaptada e nomeada VProscInsp, contendo então as etapas de planejamento, análise, coleção, discriminação, retrabalho, continuação e definindo o tipo de artefato a ser avaliado. Os resultados apontam que o uso de V&V e principalmente da técnica de Inspeção de software torna o processo de validação sistemático, de forma que os participantes sabiam o que fazer em cada fase. Contudo, o autor não deixou claro qual foi o processo de desenvolvimento em seu estudo de caso.

3.3 Trabalhos Relacionados a Testes de Software

(KOBROSLY; VASSILIADIS, 1988), em sua pesquisa sobre técnicas de teste funcional de software, nomeada "A survey of software functional testing techniques", busca

centrar-se nas metodologias sistemáticas para realizar testes e discutir os métodos de validação de software estáticos e dinâmicos. Os resultados da pesquisa, apontam a importância dessas técnicas, bem como sua utilidade no ciclo de desenvolvimento de um software. Ainda pode-se afirmar que um esforço completo de testes deve incluir diferentes técnicas, cada uma delas aplicada na fase adequada do processo de teste. Assim, o autor aborda de forma clara os tipos de testes funcionais, sua divisão e exemplifica a utilização dos mesmos.

(CRESPO ET AL; 2004) propõe em seu trabalho "Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo", um método para implantação ou melhoria do processo de teste em empresas desenvolvedoras de software. Os autores desenvolveram metodologia de teste de forma que as empresas pudessem instanciar o processo de teste de acordo com as suas necessidades e disponibilidade de recursos. Além disso, a metodologia de teste desenvolvida pode ser aplicada a qualquer tipo de software, seja ele sistema de informações ou software científico. Nesta metodologia, a implantação do processo de teste envolve um conjunto de atividades que vai desde o levantamento das necessidades da empresa, passa pela realização de treinamentos da equipe técnica e vai até ao acompanhamento dos trabalhos realizados, constituindo assim, um completo ciclo de implantação da atividade de teste dentro da empresa. Os resultados encontrados demonstraram que metodologia é viável de ser aplicada em uma micro empresa, como parte de um programa de melhoria de processo e que o processo de teste implantado pela metodologia gera melhorias visíveis aos clientes e desenvolvedores, melhorando a qualidade do software e o relacionamento entre a empresa e os clientes.

3.4 Conclusão do Capítulo

O Capítulo 3 apresentou os trabalhos relacionados, que foram selecionados pela relevância do tema abordado. Pôde-se observar nesse capítulo a utilização de algumas atividades importantes de V&V e a indicação de que técnicas podem ser utilizadas de forma individual ou em conjunto, variando de acordo com a complexidade dos sistemas desenvolvidos. Também é importante destacar que um processo independente de V&V pode reduzir custos e riscos nos projetos.

Quanto aos sistemas ERP, onde o grau de complexidade é alto e o de confiabilidade precisa ser igualmente elevado, destaca-se a importância que esses sistemas tem as organizações e como devem estar ligados as atividades de garantia de qualidade.

Os testes de software, como atividade de V&V mais utilizadas, compreendem diversos tipos e indicações de aplicação. Como as outras atividades de V&V, uma boa cobertura de testes envolve um esforço conjunto desses tipos de testes.

Na sequência, o Capítulo 4, aborda a proposta de processo elaborada com base na

revisão de literatura e nos trabalhos relacionados.

4 Processo de V&V para o NTIC

Este capítulo aborda o antigo processo de desenvolvimento utilizado pela CODEV e a proposta do novo processo. O novo processo de V&V é composto por sete atividades que são apresentadas através da notação para mapeamento do processo de trabalho – BPMN (Business Process Modeling Notation).

4.1 Processo Antigo

Como mencionado anteriormente, o processo antigo utilizado pelo NTIC continha pouca presença de atividades para garantia de qualidade e acabava sofrendo a influência direta do retrabalho.

O fluxo do processo antigo começava com a elaboração dos documentos ou requisitos vindos da CAU. Posteriormente, há a prototipação das telas e então o desenvolvedor deve analisar esses documentos e protótipos para que a implementação possa começar. Os casos de uso então são implementados e a versão finalizada é liberada para testes de aceitação. Este fluxo é representado pela Figura 3, que contém a representação modelada em BPMN.

4.2 Novo Processo Proposto

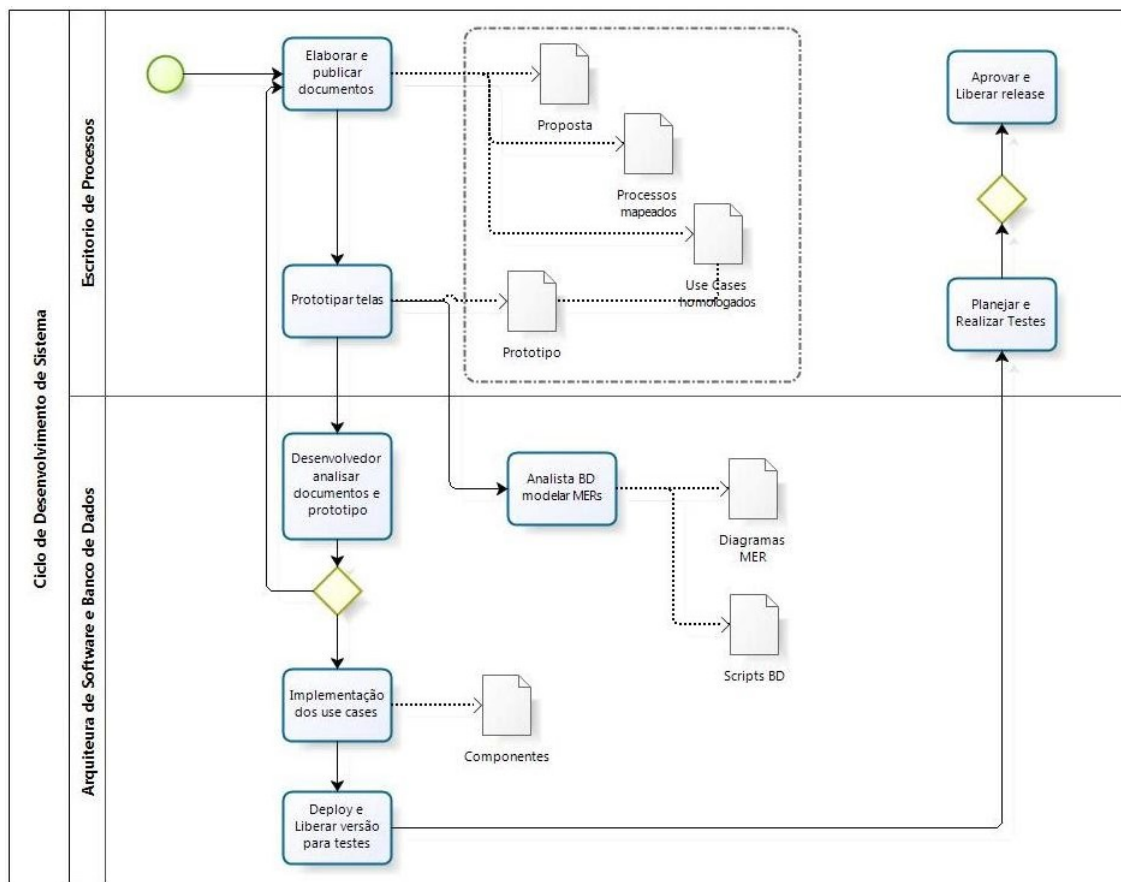
Um processo por sí só é visto como um método ou sistema para atingir algum objetivo específico. Em Engenharia de Software um processo de desenvolvimento consiste em um conjunto de atividades, ordenadas ou não, onde o objetivo principal é obter um produto de software (SOMMERVILLE, 2007).

Elaborar um processo não é uma tarefa simples, pois é necessário compreender amplamente a realidade do local que ira utilizar o processo.

Para elaborarmos a proposta do novo processo, foi necessário, além da pesquisa bibliográfica e estudo dos trabalhos relacionados, compreender a rotina de desenvolvimento da CODEV. Um dos pontos a considerar é o fato do NTIC está distribuído nos campi de Alegrete e Bagé. Sendo assim a demanda de sistemas a serem desenvolvidos pela CODEV, localizada em Alegrete, vem da CAU que está em Bagé. Além do fator de localização e conseqüentemente tempo para comunicação entre as coordenadorias é necessário considerar que a principal demanda da CODEV é manutenção e alteração de módulos já existentes.

Considerando todas as informações obtidas, o processo de V&V elaborado começa

Figura 3 – Processo de Desenvolvimento Antigo da CODEV



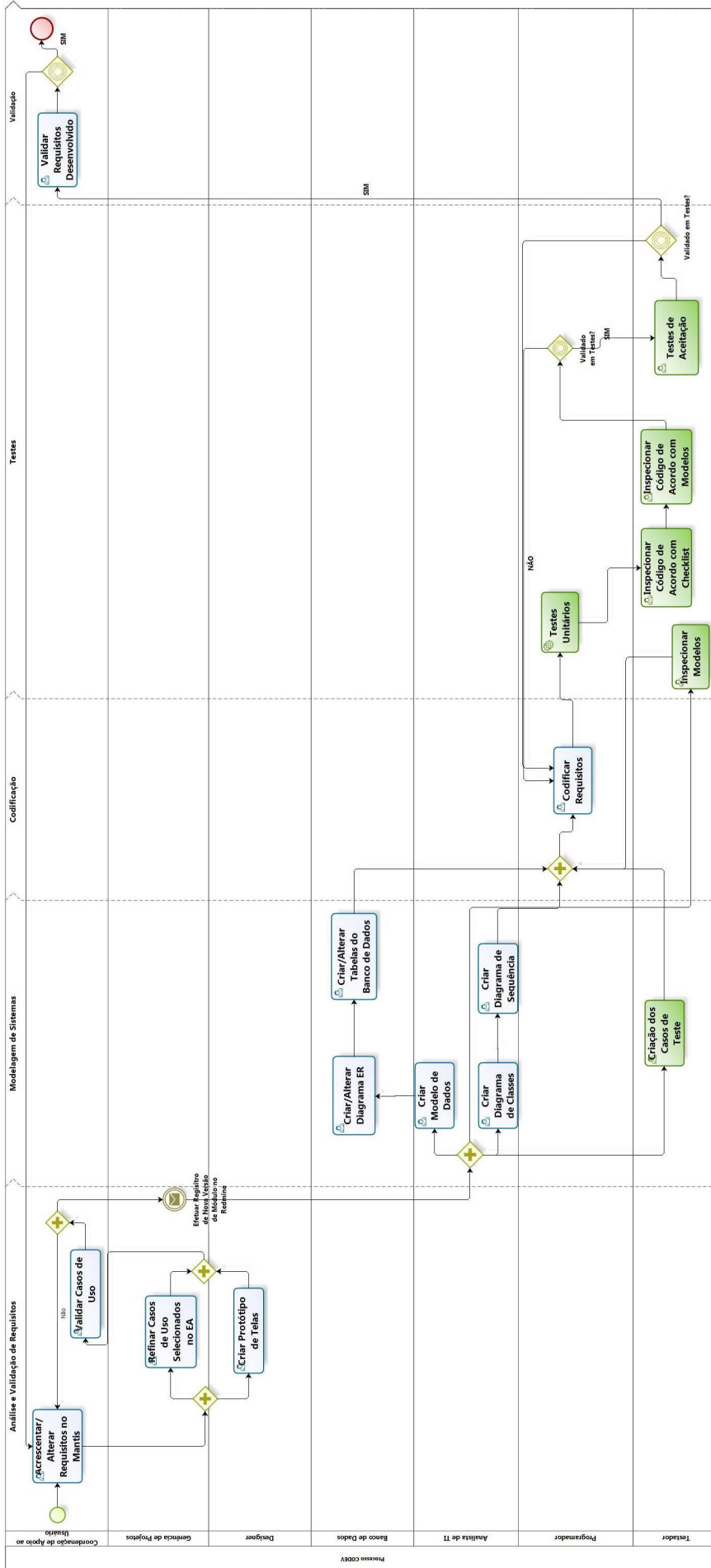
Fonte: CODEV

com a Inspeção dos requisitos, com o objetivo de refinar e validar esses requisitos enviados pela CAU. Com os requisitos validados, as próximas atividades envolvem criação dos casos de teste e inspeção dos modelos relativos ao projeto. Com os requisitos já codificados, devem ser executados os testes unitários. Posteriormente deve ser feita a inspeção do código a nível de boas práticas de programação e implementação do projeto. Com todas as etapas validadas devem ser feitos os testes de aceitação utilizando os casos de teste elaborados.

Os artefatos gerados em cada etapa auxiliam na documentação e controle das funcionalidades desenvolvidas.

A Figura 4 demonstra a modelagem completa do processo da CODEV, onde está inserido o processo de V&V que foi elaborado, utilizando a metodologia BPMN. As atividades do processo serão abordadas na sequência.

Figura 4 – Novo Processo de Desenvolvimento da CODEV



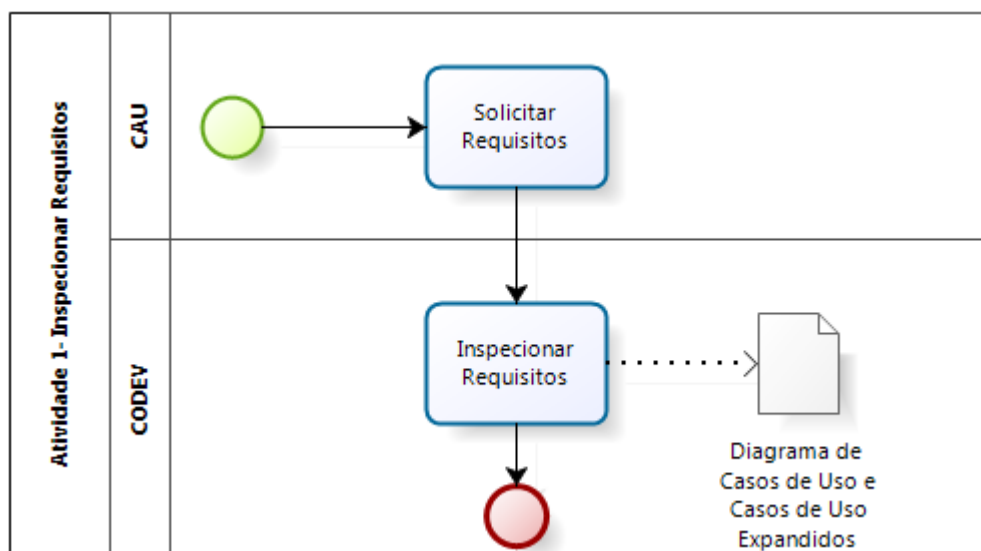
Fonte: CODEV

4.2.1 Atividade 1 - Inspeccionar requisitos

A primeira atividade do processo proposto consiste em inspecionar os requisitos que são disponibilizados pela CAU. Essa atividade tem como foco refinar e validar os requisitos solicitados, a fim de gerar maior compreensão do que se pretende desenvolver. Para validar esses requisitos é feita análise do requerimento, comunicação para esclarecimento de dúvidas e, caso necessário, uma reunião entre CAU e CODEV. A partir disso são produzidos os seguintes artefatos: diagrama de casos de uso e casos de uso expandido.

A Figura 5 demonstra o fluxo que envolve a inspeção de requisitos desde a solicitação da CAU até a geração dos artefatos.

Figura 5 – Atividade 1 - Inspeccionar Requisitos.



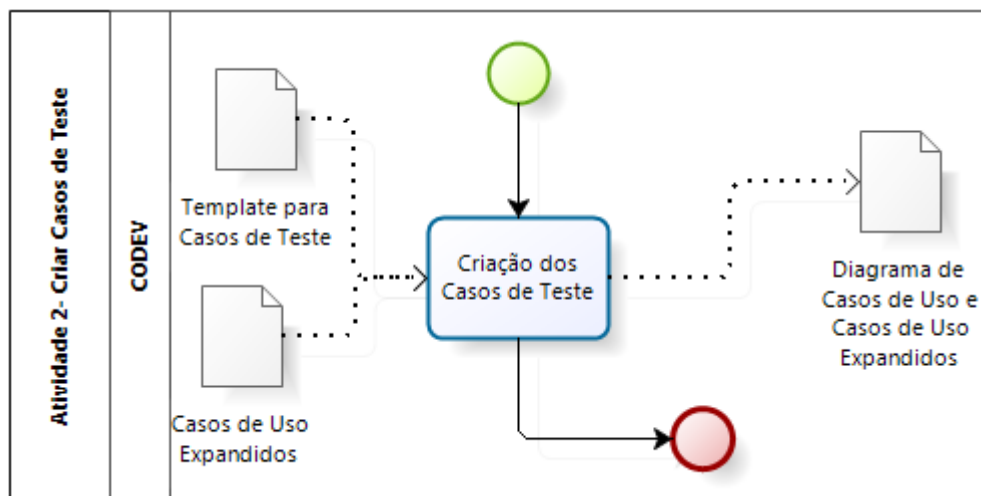
4.2.2 Atividade 2 - Criar Casos de Teste

Conforme a Figura 6, a atividade de criação dos casos de teste inicia somente após a inspeção dos requisitos e a liberação dos casos de uso expandidos associados ao mesmo. Os casos de teste são componentes fundamentais para executar o processo de testes a fim de garantir que os requisitos solicitados foram implementados corretamente e estão completos.

Para elaborar os casos de teste, é necessário adotar um padrão de passos que devem ser seguidos no momento da criação e execução do mesmo. Todos os tópicos escolhidos tem o intento de garantir maior cobertura e plenitude para os testes. O padrão adotado, compreende os seguintes tópicos: código, projeto, ID Caso de Uso, Caso de Uso, Objetivo do Teste, Pré-condições, Cenário, Entrada, Resultado esperado e Resultado obtido.

Os tópicos iniciais código, projeto, ID caso de uso e Caso de uso foram adotados para facilitar a identificação e entendimento do caso de teste, pois especificam qual o

Figura 6 – Atividade 2 - Criar Casos de Teste.



nome do projeto e caso de uso está sendo testado naquele momento.

A fim de demonstrar o comportamento da funcionalidade, as restrições necessárias e afirmar o que é esperado na execução do teste, são utilizados os tópicos de Objetivo do teste, pré-condições, cenário, entradas e resultados.

4.2.3 Atividade 3 - Inspeccionar Modelos

Conforme a Figura 7, a atividade de inspeção de modelos é executada após o finalização dos diagramas de classe, ER e sequência. Para realizar essa atividade, é utilizada a técnica de Inspeção guiada por casos de teste ou casos de uso. Essa técnica, como já abordado anteriormente, possui o diferencial de realizar a inspeção em artefatos de software de forma a avaliar se os requisitos contidos na especificação estão presentes em todos os artefatos de forma consistente, completa e correta.

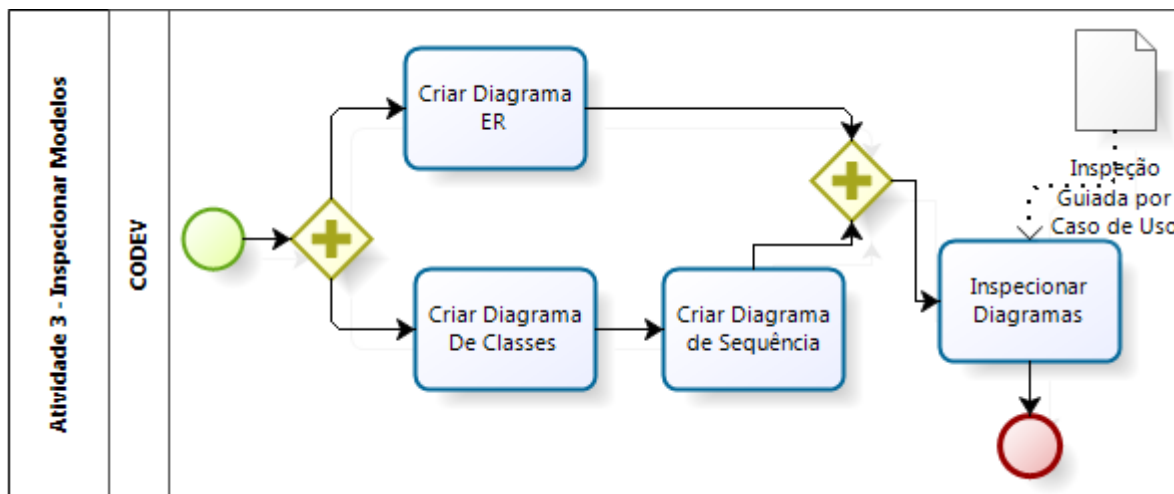
Além de utilizar uma ferramenta para essa atividade, foi definido, assim como uma atividade de inspeção de software, papéis e fluxo de execução para que a atividade fique adequada a realidade da equipe de desenvolvimento.

Em uma atividade de inspeção tradicional os papéis utilizados envolvem o autor, inspetor, relator e moderador. Para realizar essa atividade dentro do processo, customizamos os papéis para contemplar um moderador, o autor e o inspetor.

O moderador deve ser responsável por efetuar as decisões dentro da atividade, portanto deve ser alguém que compreenda com os requisitos solicitados. O autor será o responsável por desenvolver o artefato a ser inspecionado. O inspetor será responsável por avaliar o artefato desenvolvido e dispor suas considerações.

Após o autor concluir o artefato, deve comunicar ao moderador para iniciar a

Figura 7 – Atividade 3 - Inspeccionar Modelos.

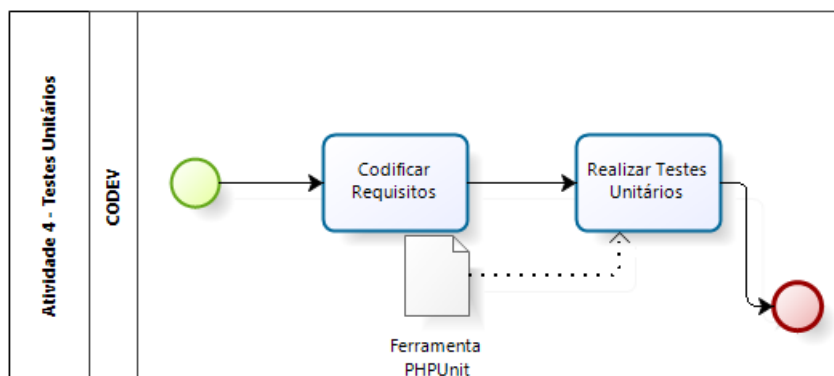


atividade de inspeção. O moderador vai definir um revisor para o artefato. O autor deve explicar e disponibilizar o artefato desenvolvido para o revisor. O revisor deve analisar o artefato e anotar possíveis alterações. Quando o revisor finalizar as considerações, deve comunicar e disponibilizar as considerações aos envolvidos. Caso existam alterações, o autor deve fazê-las. Caso existam diferentes opiniões sobre a necessidade de alterar o artefato, o moderador deve decidir. Após o autor efetuar as correções, o moderador deve decidir se há necessidade de uma nova inspeção.

4.2.4 Atividade 4 - Teste Unitário

Os testes unitários tem a função de descobrir defeitos nas menores unidades do sistema. A execução dos testes unitários fornece uma garantia ao programador de que os resultados esperados sejam iguais aos resultados retornado pelo sistema.

Figura 8 – Atividade 4- Testes Unitários.



Conforme a Figura 8, os testes unitários são realizados após a codificação dos

requisitos. Esses testes podem ser implementados pelo próprio programador da funcionalidade.

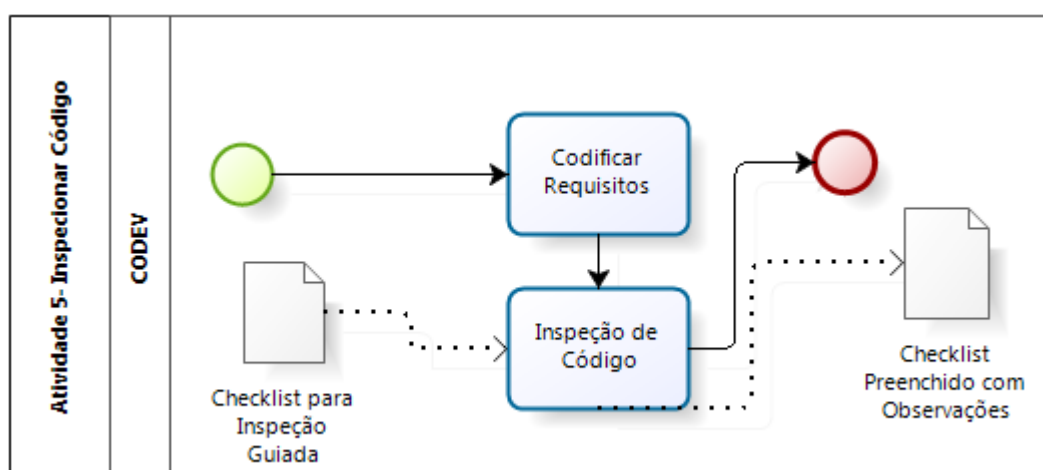
Para efetuar os testes unitários na linguagem de programação PHP, a qual é utilizada pela CODEV, é preciso dispor da ferramenta PHPUnit que auxilia na criação e execução de testes.

Após a execução dos testes unitários o programador, se considerar a funcionalidade apta, deve disponibilizar o código para as atividades de inspeção.

4.2.5 Atividade 5- Inspeccionar Código

A atividade de inspeção de código quanto a sua estrutura será guiada por um checklist já elaborado pela CODEV. O objetivo central dessa etapa é identificar se o código está dentro dos padrões solicitados e se contém boas práticas de programação.

Figura 9 – Atividade 5 - Inspeccionar a Estrutura do Código .



Conforme a Figura 9, a atividade de inspeção de código é feita após a codificação dos requisitos utilizando checklist, que é organizado em quatro grupos de informação: formatação, coerência, boas práticas e segurança. Em cada um dos grupos são definidas questões a serem respondidas pelo revisor responsável em cumprir a atividade.

Os papéis envolvidos seguem os mesmos, ou seja, tem-se o revisor, o autor e moderador atuando na execução dessa tarefa. O revisor deve ter conhecimento prévio dos itens que o checklist contempla e ao analisar o código deve completar a coluna inserindo ok ou não ok na questão avaliada. Também é possível que o revisor insira observações e dicas para o autor pode melhorar o artefato.

As atividades realizadas pelo moderador permanecem as mesmas: decidir se as modificações encontradas devem ser efetuadas, sanar as dúvidas do revisor e delegar as

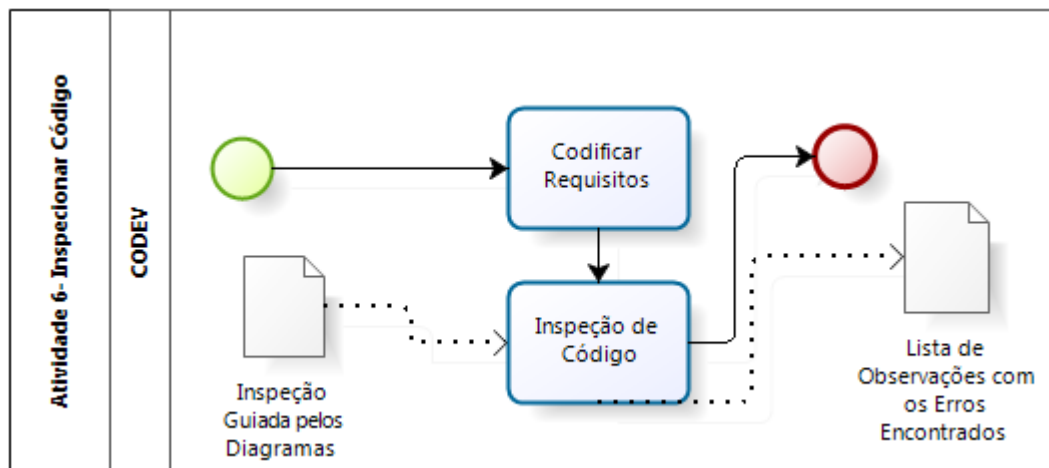
tarefas. Já as atividades do autor, como o próprio nome sugere, é desenvolver o artefato que será validado, nesse caso o próprio código.

4.2.6 Atividade 6 - Inspeccionar Código

A atividade de inspeção de código quanto a implementação do projeto, tem objetivo de verificar se a modelagem dos diagramas foi implementada corretamente na programação.

Para que essa atividade inicie, conforme a Figura 10, é necessário que o programador tenha liberado a funcionalidade para execução da inspeção, ou seja, finalizado a implementação e os testes unitários. Após isso, a inspeção será feita semelhante a anterior, onde terão papéis definidos e como ferramenta os diagramas modelados para orientar o revisor. O revisor, responsável pela realização da atividade, deve ter acesso ao código e total conhecimento da modelagem da funcionalidade. Portanto, é recomendado que o responsável pela modelagem, principalmente dos diagramas de classe e sequência, seja quem execute a atividade de inspecionar o código com relação a implementação do projeto.

Figura 10 – Atividade 6 - Inspeccionar Implementação do Projeto no Código.

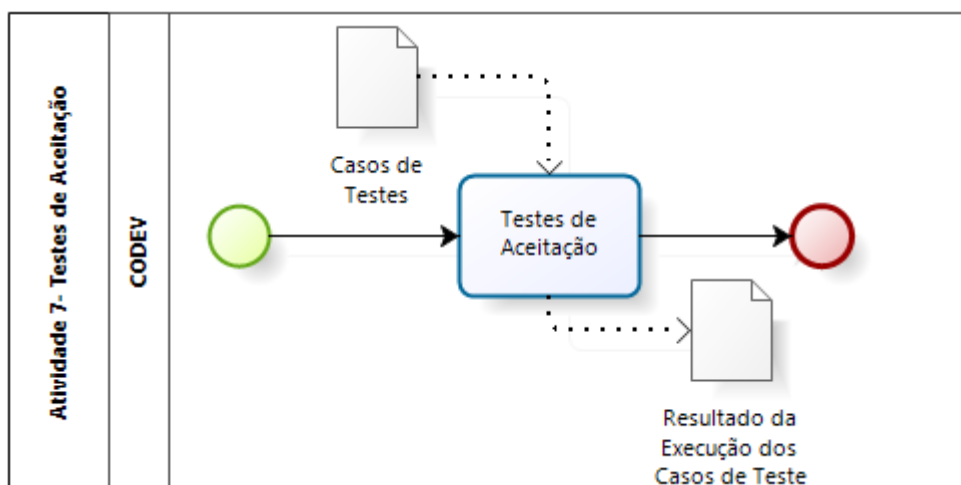


As atividades realizadas pelo moderador envolvem decidir se as modificações encontradas devem ser efetuadas, sanar as dúvidas do revisor e delegar as tarefas. Já as atividades do autor, como o próprio nome sugere, é desenvolver o artefato que será validado, nesse caso o próprio código.

4.2.7 Atividade 7- Testes de Aceitação

A etapa sete correspondente aos testes de aceitação, é realizada utilizando a técnica de inspeção guiada por casos de teste, os quais são produzidos na atividade dois. Conforme a Figura 11, com a execução dos testes de aceitação há um resultado obtido em cada caso de teste.

Figura 11 – Atividade 7- Testes de Aceitação.



Para realizar os testes de aceitação é recomendado que o responsável por executar a tarefa tenha amplo conhecimento da funcionalidade desenvolvida e siga executando os cenários descritos nos casos de teste. Com a execução dos cenários, é necessário comparar a saída encontrada com a esperada, fazendo as observações quando as mesmas forem distintas.

Caso defeitos sejam encontrados, a funcionalidade volta para correção do programador e quando finalizada são executados os testes de aceitação novamente.

4.3 Conclusão do Capítulo

O Capítulo 4 detalhou o processo de V&V que foi elaborado para o NTIC, juntamente com suas atividades e artefatos gerados.

O processo proposto é formado por sete atividades centrais ligadas a V&V: inspeção de requisitos, criação dos casos de teste, inspeção de modelos, testes unitários, duas inspeções de código e testes de aceitação.

As atividades de inspeção de modelos e código foram personalizadas para serem simples e necessitarem de poucos servidores para que sejam executadas.

O Capítulo 5, a seguir, demonstra a aplicação do processo em alguns projetos desenvolvidos na CODEV.

5 Estudo de Casos

Após a definição de processo de V&V apresentado no capítulo anterior, a aplicação desse processo dentro da CODEV ocorreu em algumas funcionalidades, sendo essas abordadas no presente capítulo.

As informações quanto a aplicação do processo estão agrupadas pelas atividades do processo que foram implementadas. As funcionalidades em que o processo foi aplicado são identificadas pelo número disponível no Mantis, ferramenta para gerenciar bugs e atividades utilizada para comunicação entre CODEV e CAU.

Para validar o processo desenvolvido, aplicamos no desenvolvimento das seguintes funcionalidades: 1529 e 1635. De forma reversa o processo foi aplicado na funcionalidade 1387, para validar a funcionalidade e verificar defeitos não descobertos pela falta de testes em seu desenvolvimento.

Na sequência são descritas as atividades do processo em cada uma das funcionalidades aplicadas. A seção 5.1 e subseções seguintes descreve a funcionalidade 1529, pacote para o módulo Portal do Aluno. A seção 5.2 e subseções aborda a funcionalidade 1635, pacote para módulo Processo Seletivo Acadêmico. A seção 5.3 relata a aplicação do processo de forma reversa na funcionalidade 1387 do módulo Plano de Trabalho, ou seja, após o desenvolvimento ter sido finalizado.

5.1 #1529 Pacote Para o Módulo PTA (Portal do Aluno)

O desenvolvimento do pacote 1529 consiste em disponibilizar no módulo portal do aluno alguns dados úteis. Os requisitos, que foram solicitados através da ferramenta Mantis pela CAU, envolvem disponibilizar nas disciplinas as quais o aluno possui matrícula o plano de ensino da disciplina e a possibilidade do aluno acompanhar o diário de classe contendo registros de presença ou faltas. A Figura 12 contém a descrição utilizada pela CAU para solicitar essas funcionalidades.

Nas subseções a seguir são abordados alguns artefatos relativos a execução das atividades do processo dentro do pacote para o módulo Portal do Aluno.

5.1.1 Inspecionar os requisitos

Com os requisitos já solicitados, a atividade de inspeção de requisitos envolve compreender, modelar e validar o que foi solicitado. A partir da análise das funcionalidades requeridas, foram desenvolvidos e encaminhados a CAU o diagrama de casos de uso e os

Figura 12 – Requisitos do pacote solicitados pela CAU.

| | |
|--|---|
| 0001529: Dados acessíveis para o aluno | |
| Descrição | <p>Solicitamos que os seguintes dados sejam disponibilizados no Portal do Aluno, para todas as disciplinas nas quais o aluno possui matrícula:</p> <ol style="list-style-type: none"> 1. Plano de Ensino da disciplina, desde que tenha sido finalizado no Portal do Professor, pelo docente. 2. Acompanhamento do diário de classe, aula a aula, desde que ela tenha sido finalizada no Portal do Professor, pelo docente. A visão dos dados do diário deve conter: data da aula, conteúdo ministrado e registro de falta do discente na aula. Os registros dos demais discentes não devem ser visíveis. <p>O Plano de Ensino deve ser visualizado em formato <u>pdf</u>. Os dados do diário podem ser visualizados em formato de relatório <u>html</u> no próprio portal.</p> |

Fonte: Ferramenta Mantis.

casos de uso expandidos para validação dos mesmos. Após aprovação da CAU é dado continuidade no processo de desenvolvimento do pacote.

A Figura 13 corresponde ao caso de uso "Visualizar plano de ensino" e contém uma breve descrição da situação, os atores envolvidos, as pré-condições exigidas na funcionalidade, o fluxo principal e alternativo de execução e as exceções que devem ser respeitadas.

Figura 13 – Caso de Uso U002- Visualizar Plano de Ensino (Aluno).

GURI-PTA-CDU-001-Visualizar Plano de Ensino (Aluno)

| | |
|----------------------------|--|
| Descrição | Este caso de uso descreve a visualização do plano de ensino de uma disciplina por parte de um aluno. |
| Ator(es) | Aluno |
| Pré-Condição | <ul style="list-style-type: none"> - Existir usuário logado. - Ser aluno. - As disciplinas em que o plano de ensino não estiver finalizado deverão aparecer para o aluno, mas não será possível visualizar o plano. - A visualização será dada em PDF. |
| Fluxo Principal | <ul style="list-style-type: none"> • P1. O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino • P2. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. • P3. Aluno imprime o PDF do plano de ensino. |
| Fluxos Alternativos | <ul style="list-style-type: none"> • A01. Aluno pesquisa plano de ensino: <ul style="list-style-type: none"> ○ A01. P1. Aluno clica no ícone busca ○ A01. P2. Aluno efetua pesquisa por ano, semestre curso e/ou disciplina. |
| Exceções | <ol style="list-style-type: none"> 1. Visualização do plano somente estará disponível se professor tiver finalizado o plano de ensino. |

5.1.2 Criar Casos de Testes

Os casos de testes foram criados após a validação dos casos de uso expandidos. Para criar os objetivos desses casos de teste é necessário avaliar as possíveis situações que a funcionalidade possa ser exposta e descrever o máximo de situações que conseguir encontrar.

Em termos quantitativos, para a funcionalidade visualizar planos de ensino foram criados 8 casos de teste e para a funcionalidade visualizar diário de classe foram criados 10 casos de teste.

A Figura 14 apresenta 2 casos de teste da funcionalidade visualizar plano de ensino contendo informações quanto ao código, projeto e caso de uso. Também são exibidos os objetivos do teste e as pré-condições exigidas para cada objetivo.

Figura 14 – Informações, Objetivos e Pré-condições dos casos de teste visualizar plano de ensino.

| ID Caso de Uso | Caso de Uso | Objetivo do teste | Pré-condições |
|------------------|------------------------------------|---|---|
| GURI-PTA-CDU-001 | Visualizar Plano de Ensino (Aluno) | Buscar Plano de Ensino por Semestre | O usuário "aluno" estar logado. |
| GURI-PTA-CDU-001 | Visualizar Plano de Ensino (Aluno) | Permitir o download somente dos planos de ensino finalizados pelo docente | O usuário "aluno" estar logado e o professor deve ter finalizado o plano de ensino. |

A Figura 15 expõe os dados que completam o caso de teste anterior, abordando o cenário para execução do caso, as entradas necessárias e o resultado que espera-se obter com sua execução.

Figura 15 – Cenário, Entradas e Resultados Esperados dos casos de teste Visualizar Plano de Ensino

| GURI Módulo Portal do Aluno | | |
|---|------------|---|
| Cenário | Entrada(s) | Resultado esperado |
| O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno . O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno clica no ícone busca, seleciona Semestre. | | O sistema retorna os planos de ensino correspondentes ao semestre pesquisado. |
| O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. O aluno clica no botão visualizar. | | O sistema retorna somente o download de planos de ensino finalizados pelo docente no Portal do Professor. |

5.1.3 Inspeccionar Modelos

A execução da inspeção dos modelos de diagrama de classes, sequência e ER foi executada pelo responsável da Atividade 1 - Inspeccionar Requisitos. O revisor relatou que o diagrama de classe não foi modelado, porém foi elaborado um modelo conceitual onde há nomes de classes e as relações entre elas. Os diagramas de sequência e modelo ER para a funcionalidade não foram implementados.

5.1.4 Inspeccionar código: estrutura

A inspeção de código guiada pelo checklist para avaliar questões de estrutura, boas práticas e outros, foi realizada por um dos analistas que auxiliou na elaboração do mesmo.

Os itens do checklist foram avaliados individualmente dentro do código fornecido pelo programador e as observações eram feitas ao lado do item analisado.

A Figura 16 demonstra o checklist utilizado no código desenvolvido para visualizar planos de ensino.

Figura 16 – Checklist para Inspeção Guiada de Código Referente a Visualizar Plano de Ensino.

| Item | Observação |
|--|---|
| 1. FORMATAÇÃO | |
| 1.1 O código-fonte está indentado e alinhado conforme o padrão? | |
| 1.2 As linhas possuem menos de 80 caracteres (se possível)? | |
| 1.3 A nomenclatura dos elementos está correta? (classes, métodos, funções, nomes de arquivos) | Procurar utilizar 3 letras para formar segmento de nomes de função (ex. getCronogramaAulaPresencial -> getCronogramaAulaNaoPresencial). |
| 1.4 A marcação xhtml está de acordo com as recomendações da W3C? | |
| 2. COERÊNCIA | |
| 2.1 As regras de negócio estão implementadas em controller/model e não na view? | |
| 2.2 Existe algum código PHP na view que possa ser movido para a controller afim de tornar a view mais limpa? | |
| 2.3 Existe algum método/função que pode ser movido para outro arquivo tendo em vista a reutilização? | Retirar funções não utilizadas que possam ter sido copiadas do esqueleto |
| 2.4 Existem trechos de código repetidos que possam ser convergidos numa função? | |
| 2.5 Nos formulários, os campos obrigatórios estão sendo validados corretamente? | |
| 2.6 Nos formulários, os campos não obrigatórios estão sendo tratados corretamente? | |
| 2.7 Na abertura das views, existe algum código "pesado" executando na controller que possa ser melhorado? exemplo: muitas consultas SQL ou consultas SQL mal formuladas ou trazendo mais resultados do que o necessário? | Verificar lentidão de acesso a dados |
| 2.8 O charset de arquivos e do banco de dados está de acordo com a programação/exibição na tela? | |
| 3. BOAS PRÁTICAS | |
| 3.1 Existe algum trecho de código com que possa ser reformulado tendo em vista a legibilidade? (com muitos ifs aninhados, laços e operadores que possam ser simplificados) | Revisar ifs aninhados e chaves (ex. linha 120 da controller). - Procurar deixar espaços em branco entre blocos para melhorar a legibilidade (entre atribuições e blocos de if, por exemplo) |
| 3.2 Existe algum trecho de código em que possa ser usado try/catch tendo em vista a legibilidade e eficiência? (exemplo: validações de edição e exclusão) | Utilizar switch na linha 142 da controller |
| 3.3 Todas as leituras de variáveis estão protegidas contra "undefined"? | |
| 3.4 O código está documentado/comentado adequadamente? | Revisar comentários de funções e cabeçalhos de arquivo (comentários copiados de outro módulo) |
| 4. SEGURANÇA | |
| 4.1 Formulários públicos que salvam dados em banco e/ou enviam e-mail estão protegidos com captcha? | Não se aplica |
| 4.2 Existem recursos restritos acessíveis diretamente via URL? | - Diminuir comprimento da URL, retirando informações que possam permitir injeção de SQL (Segurança!!) |

Um exemplo simples da aplicação do checklist é o item 3.1 pertencente a coluna de boas práticas, onde é solicitado verificar se algum trecho do código possa ser reformulado

tendo em vista melhorar a legibilidade do mesmo. O resultado encontrado no código para visualizar plano de ensino aponta que há trechos do código que podem ser reformulados pois contem alguns if's aninhados e espaços em branco desnecessários.

Com a inspeção no código concluída, o resultado foi repassado ao programador que efetuou os ajustes solicitados. A partir disso, o código passou por uma nova inspeção e obteve um resultado positivo.

5.1.5 Testes de Aceitação

A execução dos testes de aceitação foi feita pelo responsável em testes da CO-DEV, sendo guiada pelos casos de teste anteriormente elaborados. Cada caso de teste foi executado mais de uma vez para garantir que o comportamento da funcionalidade seria igual em determinado cenário. O retorno obtido com essa execução demonstrou algumas diferenças entre o resultado esperado e o obtido. Essas discrepâncias demonstraram que em alguns pontos a funcionalidade possuía defeitos e necessitaria de correções.

A Figura 17 demonstra a ocorrência de diferenças entre os resultados esperados e obtidos. Com essa diferença identificada, o caso de teste é considerado falho e precisará de correções no código e de uma reexecução dos testes de aceitação para garantir que a funcionalidade tem o comportamento esperado.

Figura 17 – Visualizar Plano de Ensino- Resultado Obtido com o teste de aceitação.

| Objetivo do teste | Resultado esperado | Resultado Obtido |
|---|---|---|
| Buscar Plano de Ensino por Semestre | O sistema retorna os planos de ensino correspondentes ao semestre pesquisado. | O sistema não efetua a busca pelo semestre. FALHOU NO TESTE |
| Permitir o download somente dos planos de ensino finalizados pelo docente | O sistema retorna somente o download de planos de ensino finalizados pelo docente no Portal do Professor. | O sistema está permitindo o download de planos de ensino não finalizados. FALHOU NO TESTE |

Com a conclusão da execução dos testes de aceitação o programador foi comunicado que correções deveriam ser feitas. Após o programador corrigir os defeitos e liberar a funcionalidade, foram reexecutados os testes de aceitação.

A figura 18 exhibe o resultado da reexecução dos testes de aceitação após a correção do programador. Os resultados obtidos com a reexecução dos casos de teste, foram iguais aos resultados esperados, ou seja, a funcionalidade foi corrigida com sucesso.

Figura 18 – Visualizar Plano de Ensino- Resultado Obtido com o reexecução dos testes.

| Objetivo do teste | Resultado esperado | Resultado Obtido na Reexecução | Obs |
|---|---|---|-----|
| Buscar Plano de Ensino por Semestre | O sistema retorna os planos de ensino correspondentes ao semestre pesquisado. | O sistema está efetuando a busca correta pelo semestre | OK |
| Permitir o download somente dos planos de ensino finalizados pelo docente | O sistema retorna somente o download de planos de ensino finalizados pelo docente no Portal do Professor. | O sistema está retornando somente os planos de ensino finalizados pelo docente. | OK |

5.2 #1635 Pacote Para o Módulo PSA (Processo Seletivo Acadêmico)

O desenvolvimento do pacote 1635 consiste em disponibilizar no módulo Processo Seletivo Acadêmico a possibilidade de anexar arquivos durante a inscrição de um candidato. Os requisitos, que foram solicitados através da ferramenta Mantis pela CAU, envolvem disponibilizar a função de habilitar o envio de arquivos durante a inscrição, o candidato anexar arquivos e a visualização desses arquivos enviados pelo candidato e pela administração do processo.

Nas subseções a seguir são apresentados alguns dos artefatos que foram obtidos com a execução das atividades relativas ao processo dentro do pacote para o módulo Processo Seletivo Acadêmico.

5.2.1 Inspeccionar os requisitos

A atividade de inspeção de requisitos envolveu validação e modelagem das funcionalidades solicitadas. A partir disso, foram desenvolvidos e encaminhados a CAU o diagrama de casos de uso e os casos de uso expandidos para validação dos mesmos. Após aprovação da CAU é dado continuidade no processo de desenvolvimento do pacote.

Os seguintes casos de uso foram gerados: Habilitar Anexos de Arquivos Durante a Inscrição, Anexar Arquivos, Visualizar Arquivos Enviados, Visualizar Arquivos Enviados (Administração).

A Figura 19 corresponde ao caso de uso Anexar Arquivos e contém uma breve descrição relacionada ao anexo de arquivos no momento da inscrição em processos seletivos, bem como do candidato com ator envolvido, as pré-condições exigidas, o fluxo principal e alternativo de execução e as exceções que devem ser respeitadas.

Figura 19 – Caso de Uso U002- Anexar Arquivos.

GURI-PSA-CDU-002-Anexar Arquivos

| | |
|-----------------------------|---|
| Descrição/Requisitos | Anexar arquivos durante inscrição de processos seletivos acadêmicos |
| Ator (es) | Candidato |
| Pré-Condição | 1. Envio de arquivos habilitado via configuração do processo seletivo (CDU-001). |
| Fluxo Principal | P1. Usuário acessa Acadêmico -> Processo Seletivo -> Nova Inscrição; P2. Sistema exibe processos seletivos cadastrados; P3. Usuário clica no processo seletivo desejado; P4. Sistema exibe acesso à área do candidato; P5. Usuário preenche todos os dados; P6. Se configuração estiver habilitada para envio de arquivos, sistema exibe modal para emissão de arquivos no formato PDF; P7. Usuário envia arquivos desejados e salva; P8. Sistema exibe área do candidato. |
| Exceções | 1. Somente será possível adicionar arquivos do tipo PDF. 2. Usuário poderá adicionar quantos arquivos quiser. |

5.2.2 Criar Casos de Testes

Com a validação dos casos de uso por parte da CAU, os casos de teste foram elaborados visando uma boa cobertura das situações que poderiam ocorrer.

Em termos quantitativos, para a funcionalidade habilitar o envio de anexos foram criados 3 casos de teste, anexar arquivos obteve 4 casos de teste e visualizar arquivos anexados quatro 4 casos de teste.

A Figura 20 demonstra um caso de teste da funcionalidade Anexar Arquivos contendo informações quanto ao código, projeto e caso de uso.

Figura 20 – Informações quanto aos Casos de teste Manter PSA.

| Código | Projeto | ID Caso de Uso | Caso de Uso |
|---------|--|------------------|----------------------------|
| 0001635 | GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-001 | Manter Processos Seletivos |

A Figura 21 apresenta as informações complementares ao caso de teste anterior, como o objetivo do teste, as pré-condições e cenário descrito.

A Figura 22 exibe a última informação descrita no caso de teste: o resultado esperado. Tendo como o objetivo do teste anexar vários arquivos, o resultado deve contemplar o sistema salvando os arquivos anexados e a inscrição do candidato.

Figura 21 – Caso de Teste Manter PSA- Continuação.

| Objetivo do teste | Pré-condições | Cenário | Entrada(s) |
|---|--|---|-------------|
| Anexar vários arquivos durante a inscrição do PSA | O usuário deve se inscrever como "Candidato". A opção de upload de arquivos deve estar habilitada. | Usuário acessa o menu Acadêmico -> Processo Seletivo -> Nova Inscrição. O Usuário clica no processo seletivo desejado e preenche todos os dados solicitados. O usuário clica na opção para fazer o upload de arquivos, clica no botão "Procurar", seleciona um arquivo PDF, clica no botão "Adicionar Arquivo", repetir a operação até adicionar os arquivos desejados, clica no botão "Anexar Arquivos" e clica no botão "Salvar". | Arquivo PDF |

Figura 22 – Caso de teste Manter PSA- Resultados Esperados.

| Resultado esperado |
|---|
| O sistema deve salvar o arquivo anexado juntamente com inscrição do candidato, exibir um feedback ao usuário (mensagem) e redirecionar à área do candidato. |

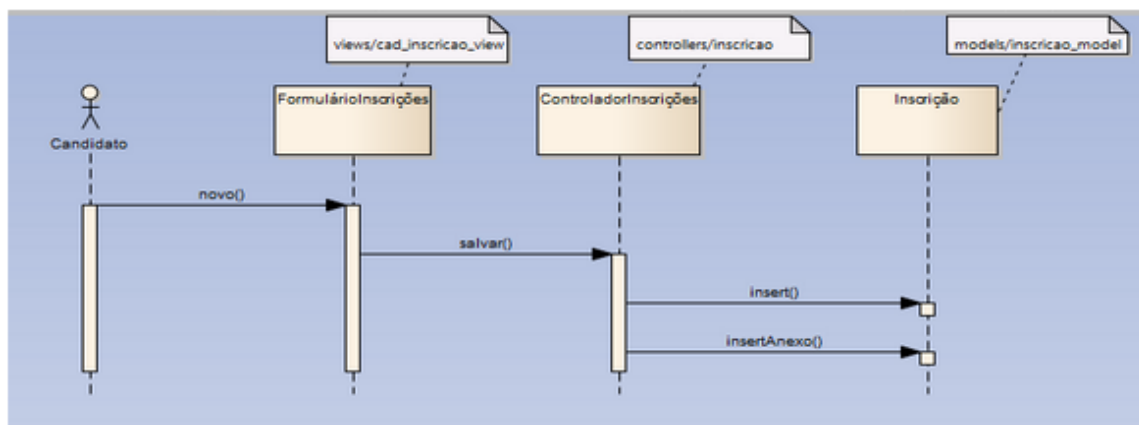
5.2.3 Inspeccionar Modelos

A execução da inspeção dos modelos de diagrama de classes, sequência e ER também foi executada pelo responsável da Atividade 1 - Inspeccionar Requisitos. O revisor relatou que o diagrama de classe não foi modelado, porém foi elaborado um modelo conceitual onde há nomes de classes e as relações entre elas.

Os diagramas de sequência modelados correspondem as funcionalidades Anexar Arquivos e Visualizar Arquivos Anexados, ambos sob perspectiva do candidato. De acordo com o revisor da inspeção, os diagramas foram considerados incompletos sob duas perspectivas. A primeira tem como justificativa os diagramas não contemplarem nenhum tipo de retorno. Já a segunda se dá pelo fato de estarem faltando dois diagramas que seriam correspondentes as funcionalidades de habilitar o anexo de arquivos e visualizar os arquivos anexados como administrador.

A Figura 23 demonstra o diagrama de sequência da funcionalidade Anexar Arquivos considerado incompleto pelo revisor. No diagrama está modelado a sequência que deveria ser implementada para anexar arquivos, onde demonstra somente o ator "Candidato" criando uma nova inscrição, salvando-a e utilizando um método *insert()* para anexar o arquivo.

Figura 23 – Diagrama de Sequência Anexar Arquivos



5.2.4 Inspeccionar código: estrutura

A inspeção de código guiada pelo checklist para avaliar questões de estrutura, boas práticas e outros, também foi realizada por um dos analistas que auxiliou na elaboração do mesmo.

Os itens do checklist foram avaliados individualmente dentro do código fornecido pelo programador e as observações eram feitas ao lado do item analisado. Diferente da inspeção de código do 1529 que resultou em dois checklists, um para cada caso de uso, essa inspeção foi elaborada em somente um checklist pois os métodos eram simples e relativamente pequenos.

A Figura 24 demonstra o checklist utilizado no código desenvolvido para o pacote 1635.

Exemplificando a aplicação do checklist é o item 3.4 pertencente a coluna de boas práticas, onde é solicitado se o código está comentado e documentado corretamente. O resultado encontrado aponta que diversas funções não possuem comentários indicando o que ela faz, ou seja, grande parte das funções não possuem uma documentação adequada.

5.2.5 Testes de Aceitação

A execução dos testes de aceitação também foi realizada pelo responsável em testes da CODEV, sendo guiada pelos casos de teste elaborados na Atividade 2. Os casos de testes foram executados mais de uma vez para garantir que o comportamento da funcionalidade seria uniforme em determinados cenários. O retorno obtido com essa execução apontou algumas diferenças entre o resultado esperado e o obtido, demonstrando que em alguns pontos a funcionalidade possuía defeitos e necessitava de correções.

Figura 24 – Checklist Utilizado no Pacote 1387.

| Item | Observação |
|--|--|
| 1. FORMATAÇÃO | |
| 1.1 O código-fonte está indentado e alinhado conforme o padrão? | Rever o alinhamento do código e tabulações |
| 1.2 As linhas possuem menos de 80 caracteres (se possível)? | |
| 1.3 A nomenclatura dos elementos está correta? (classes, métodos, funções, nomes de arquivos) | |
| 1.4 A marcação xhtml está de acordo com as recomendações da W3C? | Não se aplica - checklist apenas de código |
| 2. COERÊNCIA | |
| 2.1 As regras de negócio estão implementadas em controller/model e não na view? | |
| 2.2 Existe algum código PHP na view que possa ser movido para a controller afim de tornar a view mais limpa? | |
| 2.3 Existe algum método/função que pode ser movido para outro arquivo tendo em vista a reutilização? | |
| 2.4 Existem trechos de código repetidos que possam ser convergidos numa função? | |
| 2.5 Nos formulários, os campos obrigatórios estão sendo validados corretamente? | |
| 2.6 Nos formulários, os campos não obrigatórios estão sendo tratados corretamente? | |
| 2.7 Na abertura das views, existe algum código "pesado" executando na controller que possa ser melhorado? exemplo: muitas consultas SQL ou consultas SQL mal formuladas ou trazendo mais resultados do que o necessário? | |
| 2.8 O charset de arquivos e do banco de dados está de acordo com a programação/exibição na tela? | |
| 3. BOAS PRÁTICAS | |
| 3.1 Existe algum trecho de código com que possa ser reformulado tendo em vista a legibilidade? (com muitos ifs aninhados, laços e operadores que possam ser simplificados) | Melhorar alinhamento e tabulação, espaços entre blocos |
| 3.2 Existe algum trecho de código em que possa ser usado try/catch tendo em vista a legibilidade e eficiência? (exemplo: validações de edição e exclusão) | |
| 3.3 Todas as leituras de variáveis estão protegidas contra "undefined"? | |
| 3.4 O código está documentado/comentado adequadamente? | Faltam comentários em diversas funções |
| 4. SEGURANÇA | |
| 4.1 Formulários públicos que salvam dados em banco e/ou enviam e-mail estão protegidos com captcha? | |
| 4.2 Existem recursos restritos acessíveis diretamente via URL? | |

A Figura 25 demonstra a ocorrência de diferenças entre os resultados esperados e obtidos. Com essa diferença identificada, o caso de teste é considerado falho e precisará de correções no código.

Figura 25 – Resultado da execução do Caso de teste Anexar Arquivos.

| Resultado esperado | Resultado Obtido | Observações |
|---|--|-------------|
| O sistema deve salvar o arquivo anexado juntamente com inscrição do candidato, exibir um feedback ao usuário (mensagem) e redirecionar à área do candidato. | O sistema não está mostrando todos os arquivos anexados. FALHOU NO TESTE | Não |

A execução dos casos de teste foi compartilhada com o programador que efetuou as correções necessárias e obteve resultado positivo na reexecução dos testes na funcionalidade.

5.3 #1387 Pacote para o Módulo PTR (Plano de Trabalho)

O desenvolvimento do pacote 1387 consiste em disponibilizar o módulo Plano de Trabalho. Os requisitos, que foram solicitados através da ferramenta Mantis pela CAU, envolvem manter o plano de trabalho (inserir, editar, copiar, enviar), visualizar e inserir observações no plano.

O processo de V&V não foi aplicado na construção das funcionalidades e sim em modo reverso, com objetivo de descobrir se há alguma inconsistência no módulo por conseguinte de seu desenvolvimento não contemplar atividades para garantia de qualidade.

5.3.1 Inspeccionar os requisitos

A atividade de inspeção de requisitos foi a única atividade do processo de V&V realizada no decorrer do desenvolvimento, tendo em vista a complexidade do módulo solicitado. Com a análise das funcionalidades requeridas, foram desenvolvidos e encaminhados a CAU o diagrama de casos de uso e os casos de uso expandidos para validação. Essa validação com a CAU durou aproximadamente 7 dias, onde ocorreram mudanças nos requisitos e atualização dos artefatos.

Após concluir a inspeção os seguintes casos de uso foram gerados: "Manter plano de trabalho", "Visualizar plano de trabalho" e "Inserir observações no plano de trabalho".

Figura 26 – Caso de Uso 004- Inserir Observações no Plano de Trabalho(chefia imediata).

GURI-PTR-CDU-004-Inserir Observações no Plano de Trabalho (chefia imediata)

| | |
|----------------------------|--|
| Descrição | A chefia somente adiciona informações ao plano de trabalho do subordinado. Todas as observações devem ficar salvas e não podem ser apagadas. A chefia somente devolve ao servidor uma vez, situação NÃO REFERENCIADA (pendente de ajustes). Na segunda passada pela chefia fica na situação REFERENCIADA (aprovada). O plano ficará aberto até o final do ano/referência para acréscimo de informações. |
| Ator (es) | Servidor (chefe) |
| Pré-Condição | Servidor logado; Plano de trabalho inserido com situação 2 ou 4; |
| Fluxo Principal | <ul style="list-style-type: none"> • P1. O Servidor entra na funcionalidade; • P2. O sistema verifica se o servidor logado é chefe de outro servidor e lista os planos na situação "2" ou "4"; Ver TELA 03; • P3. O Servidor seleciona o plano "2"; • P4. O sistema busca as informações e apresenta; • P5. O Servidor confere e aprova o plano; Ver TELA 04; • P6. O Sistema seta a situação "5" e armazena as informações. |
| Fluxos Alternativos | <ul style="list-style-type: none"> • A01. P5. O Servidor confere e não aprova o plano: <ul style="list-style-type: none"> ◦ A01. P5. a. O sistema mostra opção de inserir observações por categoria; ◦ A01. P5. b. O Servidor insere observações e salva; Ver TELA 05; ◦ A01. P5. c. O sistema seta a situação "3" e salva informações e retorna ao passo P2 do fluxo principal; • A02. P3. O Servidor seleciona o plano "4"; <ul style="list-style-type: none"> ◦ A01. P5. a. O sistema busca as informações e apresenta; ◦ A01. P5. b. O Servidor aprova; Ver TELA 05; ◦ A01. P5. c. O sistema seta a situação "5" e salva informações e retorna ao passo P2 do fluxo principal; |
| Exceções | |

A Figura 26 mostra o caso de uso "Inserir observações no plano de trabalho" que contém descrição geral e informações quanto ao caso de uso.

5.3.2 Criar Casos de Testes

Os casos de teste foram criados com base nos artefatos disponíveis (casos de uso expandidos).

Em termos quantitativos, para o caso de uso manter planos de trabalho foram criados 11 casos de teste, visualizar plano de trabalho dispõe de dois 2 casos de teste e inserir observações no plano de trabalho oito 8 casos de teste.

A Figura 27 exibe um caso de teste da funcionalidade "Inserir observações no plano de trabalho" que contém informações quanto ao código, projeto, ID e caso de uso.

Figura 27 – Informações do Caso de Teste Inserir Observações no Plano de Trabalho.

| Código | Projeto | ID Caso de Uso | Caso de Uso |
|--------|--------------------------------|----------------|--|
| 1328 | GURI - PTR - Plano de Trabalho | CDU-004 | Inserir Observações no Plano de Trabalho (chefia imediata) |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-004 | Inserir Observações no Plano de Trabalho (chefia imediata) |

A Figura 28 apresenta as informações relativas ao caso de teste anterior, como o objetivo do teste, as pré-condições e cenário a ser executado. Os objetivos do teste correspondem as situação de não aprovar o plano e de aprovar o plano e inserir observações. Os cenários descritos devem conduzir a execução para esses dois objetivos.

Figura 28 – Caso de Teste Inserir Observações no Plano de Trabalho - Continuação.

| Objetivo do teste | Pré-condições | Cenário |
|---------------------------------------|--|--|
| Não aprovar o plano | O usuário "Servidor" deve estar logado e ser chefia. | O usuário clica no Menu Administrativo -> Recursos Humanos-> Plano de Trabalho-> Observações-Listar Planos. O usuário clica em visualizar na linha correspondente ao plano que deseja. O usuário clica no botão para adicionar observações e atividades se necessário. |
| Aprovar o plano e inserir observações | O usuário "Servidor" deve estar logado e ser chefia. | O usuário clica no Menu Administrativo -> Recursos Humanos-> Plano de Trabalho-> Observações-Listar Planos. O usuário clica em visualizar na linha correspondente ao plano que deseja. O usuário clica no botão para adicionar observações e atividades se necessário. |

A Figura 29 exibe o restante do caso de teste: as entradas e o resultado esperado. Os resultados esperados devem condizer com o objetivo do teste, que nesse caso é não aprovar e aprovar o plano. Esses resultados esperados serviram para analisar as saídas obtidas na execução dos testes de aceitação.

Figura 29 – Casos de teste - Entrada e Resultados Esperados.

| Entrada(s) | Resultado esperado |
|------------|---|
| | O sistema salva as observações inseridas pela Chefia e troca a situação do plano para "Não referendada". |
| | O sistema deve salvar as observações e trocar o status do plano para referendado. O sistema deve exibir as observações ao servidor. |

5.3.3 Inspeccionar Modelos

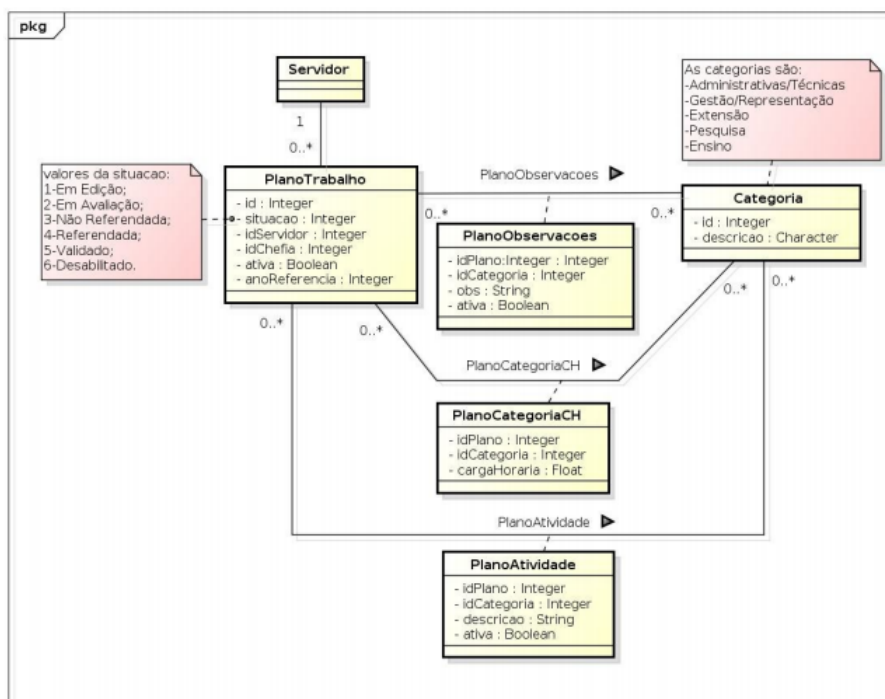
Para as funcionalidades do plano de trabalho, foram modelados os diagramas de classe e banco de dados, portanto não há diagramas de sequência.

O revisor destacou que o diagrama de classes modelado está correto, porém incompleto. O diagrama está contemplando as classes, atributos e relações, entretanto não há nenhum método modelado.

Para o banco de dados não há um diagrama de entidade-relacionamento porém foi modelado tabelas necessárias, atributos, tipos, tamanhos, valores e se os atributos são obrigatórias ou não.

A Figura 30 demonstra o diagrama de classes que foi elaborado contendo somente, além das classes, os relacionamentos e atributos.

Figura 30 – Diagrama de classes modelado para o plano de trabalho.



5.3.4 Inspeccionar código: estrutura

A inspeção de código também foi guiada pelo checklist para avaliar questões de estrutura, boas práticas e outros. Como executado nas outras inspeções, os itens do checklist foram avaliados individualmente dentro do código fornecido pelo programador e as observações foram feitas ao lado do item analisado.

A Figura 31 demonstra o checklist utilizado no código desenvolvido para o pacote 1387.

Figura 31 – Checklist utilizado no código do módulo Plano de Trabalho.

| Item | Observação |
|--|---|
| 1. FORMATAÇÃO | |
| 1.1 O código-fonte está indentado e alinhado conforme o padrão? | Colocar as tabulações e alinhamento conforme o padrão; Retirar linhas em branco "perdidas" e melhorar a legibilidade |
| 1.2 As linhas possuem menos de 80 caracteres (se possível)? | |
| 1.3 A nomenclatura dos elementos está correta? (classes, métodos, funções, nomes de arquivos) | |
| 1.4 A marcação xhtml está de acordo com as recomendações da W3C? | Revisar listagem e formulário de plano de trabalho do servidor; Visualização de plano de trabalho. Revisar IDs de elementos HTML |
| 2. COERÊNCIA | |
| 2.1 As regras de negócio estão implementadas em controller/model e não na view? | Verificar valores de situação colocados diretamente na controller (plano_trabalho). Devem ser colocados como constantes, se for o caso, no início do arquivo ou globalmente |
| 2.2 Existe algum código PHP na view que possa ser movido para a controller afim de tornar a view mais limpa? | |
| 2.3 Existe algum método/função que pode ser movido para outro arquivo tendo em vista a reutilização? | |
| 2.4 Existem trechos de código repetidos que possam ser convergidos numa função? | |
| 2.5 Nos formulários, os campos obrigatórios estão sendo validados corretamente? | |
| 2.6 Nos formulários, os campos não obrigatórios estão sendo tratados corretamente? | |
| 2.7 Na abertura das views, existe algum código "pesado" executando na controller que possa ser melhorado? exemplo: muitas consultas SQL ou consultas SQL mal formuladas ou trazendo mais resultados do que o necessário? | |
| 2.8 O charset de arquivos e do banco de dados está de acordo com a programação/exibição na tela? | |
| 3. BOAS PRÁTICAS | |
| 3.1 Existe algum trecho de código com que possa ser reformulado tendo em vista a legibilidade e eficiência? (com muitos ifs aninhados, laços e operadores que possam ser simplificados) | Verificar ifs aninhados em controllers |
| 3.2 Existe algum trecho de código em que possa ser usado try/catch tendo em vista a legibilidade e eficiência? (exemplo: validações de edição e exclusão) | |
| 3.3 Todas as leituras de variáveis estão protegidas contra "undefined"? | |
| 3.4 O código está documentado/comentado adequadamente? | |
| 4. SEGURANÇA | |
| 4.1 Formulários públicos que salvam dados em banco e/ou enviam e-mail estão protegidos com captcha? | |
| 4.2 Existem recursos restritos acessíveis diretamente via URL? | |

Em uma das respostas obtidas com o checklist é o item 2.1 pertencente a coluna de coerência, onde é solicitado se as regras de negócio estão sendo implementadas na model e não na view. O resultado encontrado aponta que diversas funções não possuem comentários indicando o que ela faz, ou seja, grande parte das funções não possuem uma documentação adequada.

5.3.5 Testes de Aceitação

Os casos de testes foram executados várias vezes para garantir que o comportamento da funcionalidade seria uniforme nos cenários descritos. O retorno obtido com essa execução apontou algumas diferenças entre o resultado esperado e o obtido, demonstrando que em alguns pontos a funcionalidade possuía defeitos e necessitava de correções.

A Figura 32 demonstra a ocorrência de diferenças entre os resultados esperados e obtidos. Com essa diferença identificada, o caso de teste é considerado falho e é indicado que sejam feitas as correções. O programador responsável foi avisado quanto a necessidade de correção nas funcionalidades.

Figura 32 – Testes de Aceitação Referente ao Caso de Uso Inserir Observações no Plano de Trabalho.

| Resultado esperado | Resultado Obtido | Observações |
|---|--|-------------|
| O sistema salva as observações inseridas pela Chefia e troca a situação do plano para "Não referendada". | O sistema está duplicando a observação quando a situação foi não referendada. FALHOU NO TESTE. | NÃO |
| O sistema deve salvar as observações e trocar o status do plano para referendado. O sistema deve exibir as observações ao servidor. | O sistema não está exibindo as informações quando o plano é aprovado. FALHOU NO TESTE. | NÃO |

5.4 Conclusão do Capítulo

O Capítulo 5, demonstrou alguns dos resultados obtidos com a aplicação do processo. Sendo utilizado normalmente ou reversamente, o processo demonstrou muita utilidade em descobrir defeitos no sistema. Esses defeitos foram descobertos tanto nas atividades de testes, quanto nas atividades de inspeção de código e modelos.

O processo proposto foi aplicado em 3 pacotes dos módulos Portal do Aluno, Processo Seletivo Acadêmico e Plano de Trabalho. Esses pacotes contêm poucas funcionalidades, permitindo a realização de um bom estudo de casos.

O Capítulo 6 aborda na sequência os resultados encontrados com esse estudo de casos.

6 Resultados

Após a aplicação do processo proposto nos projetos citados no Capítulo 5, foram coletados os dados obtidos para que fosse realizada a interpretação dos mesmos. Com a análise preliminar, foi possível verificar as mudanças positivas no desenvolvimento das funcionalidades, bem como a diminuição do retrabalho pelo aumento da garantia de qualidade no processo.

Como principal indicador do processo de V&V, o retrabalho foi medido através de comparação entre a quantidade de defeitos descobertos pelo usuário/cliente (aceitação e implantação) em pacotes gerados sem o processo e com o processo. Também caracterizamos os erros pelo momento em que foram encontrados: antes, ou seja, durante o desenvolvimento das funcionalidade e depois, quando o desenvolvimento estiver concluído.

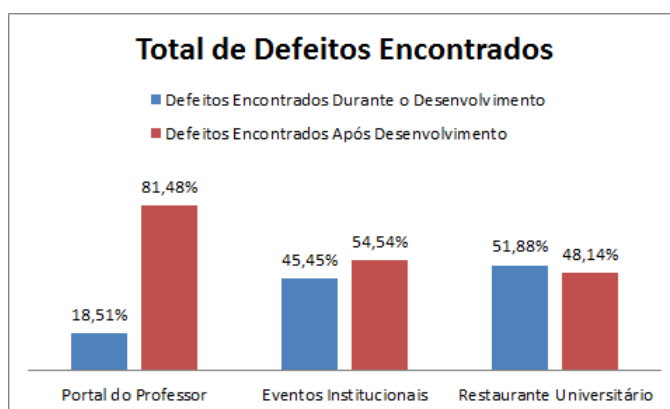
Ainda quanto ao retrabalho, foi analisado os custos envolvidos para o desenvolvimento das funcionalidades e os gerados pelo mesmo. Para estimar o custo, foi necessário calcular as horas gastas na produção das funcionalidade, levando em conta o esforço dedicado em especificação, implementação e testes, o valor pago por hora ao servidores envolvidos e as horas utilizadas para correção das funcionalidades.

O cálculo utilizado para determinar o custo das horas do servidores, foi realizado a partir do salário recebido, dividido pela quantidade de horas mensais trabalhadas. O salário básico de analista na CODEV é de R\$ 3.392,00 e para o técnico é de R\$2.115,00, correspondentes a 40 horas semanais ou 160 mensais. A partir disso o valor por hora do analista custa aproximadamente R\$ 21,20 e R\$ 13,20 do técnico. Quando há interação do técnico e analista, como por exemplo no esforço para programação, é calculada a média aritmética entre os salários resultando no valor de R\$ 17,20 a hora.

A Figura 33 apresenta o módulo "PTP -Portal do Professor", "Eventos Institucionais" e "Restaurante Universitário". Para a funcionalidade "Plano de Ensino" do módulo PTP, alterações no módulo de "Eventos" e criação do módulo "RU", ambas sem a aplicação do processo de V&V.

Os defeitos encontrados na funcionalidade do "PTP", 27 no total, foram agrupados pelo momento em que foram encontrados, antes do processo terminar foram encontrados 5 defeitos e 22 depois do processo ser concluído, sendo assim 18,51% dos defeitos são descobertos durante o processo e 81,49% pelos usuários utilizando a funcionalidade. Os 81 defeitos nas alterações do módulo "EVT", foram encontrados da seguinte forma: 42 durante o processo de desenvolvimento e 39 após o processo concluído. Portanto 51,85% dos defeitos foram descobertos antes e 48,14% depois. Já os defeitos encontrados no módulo "RU", foram encontrados 5 durante o processo de desenvolvimento e 6 após o processo

Figura 33 – PTP , EVT e RU - Total de Defeitos Encontrados .

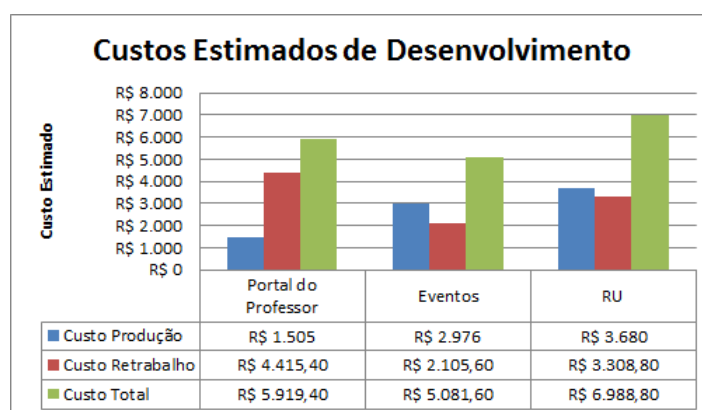


Fonte:Elaborado Pelo Autor.

de desenvolvimento, totalizando 45,45% descobertos antes e 54,54% depois. A partir desses dados podemos concluir que 2/3 dos módulos os defeitos foram encontrados após o desenvolvimento, caracterizando retrabalho para correção desses defeitos encontrados.

A Figura 34 exibe o custo estimado de desenvolvimento das funcionalidades dos módulos apresentados: PTP, EVT e RU. O custo aproximado para desenvolvimento das funcionalidade para o módulo "PTP"totalizaram R\$5.919,40, sendo 74,59% custos de retrabalho e 24,41% de produção. Para as alterações no módulo "EVT", os custos estimados obtiveram um totalde R\$5.081,60, onde 52,66% eram oriundos da produção e 47,34% do retrabalho. Quanto ao módulo "RU", o gasto total foi de R\$6.988,80, onde 58,57% eram gastos de produção e 41,43% de correção. Sendo assim em 1/3 dos módulos o gasto aproximado com o retrabalho é superior ao gasto obtido com produção. Ainda é possível perceber que em 2/3 dos módulos o custo aproximado de produção e retrabalho possui uma diferença de menos de R\$900.

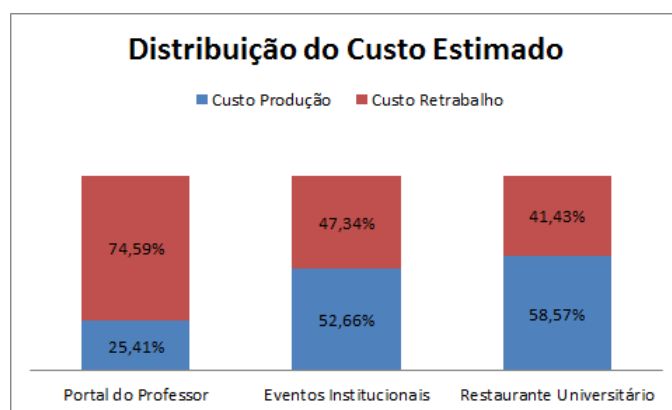
Figura 34 – PTP , EVT e RU - Custo Estimado de Desenvolvimento.



Fonte:Elaborado Pelo Autor.

A Figura 35 está relacionada com a distribuição dos custos estimados de desenvolvimento. Como já mencionado, ela retrata em porcentagem a quantidade de investimentos utilizados para desenvolvimento e correção das funcionalidades. Podemos perceber que em 1/3 dos módulos o custo de retrabalho é superior ao de produção e em 2/3 dos módulos custo de retrabalho e produção não difere muito.

Figura 35 – PTP , EVT e RU - Distribuição do Custo Estimado de Desenvolvimento .



Fonte:Elaborado Pelo Autor.

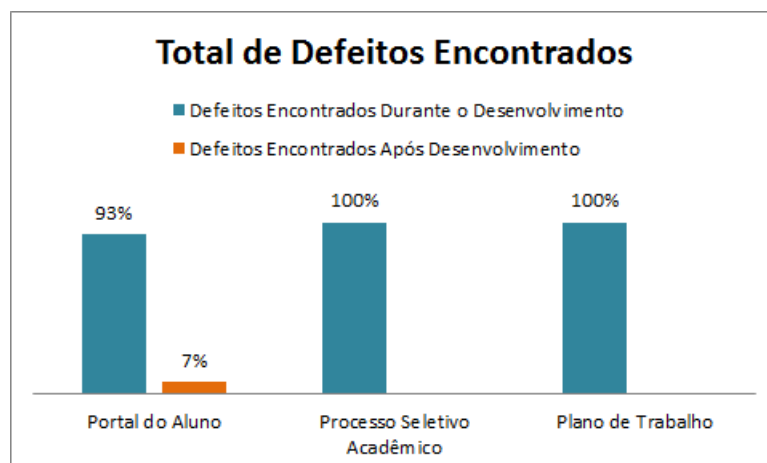
Com a aplicação do processo, os resultados encontrados contrastam com os anteriores. O processo aplicado nas funcionalidades do "PTA Portal do Aluno", "PSA- Processo Seletivo Acadêmico" e "PTR - Plano de Trabalho", trazem resultados positivos quanto a diminuições de custos por retrabalho.

A Figura 36 consiste no gráfico de defeitos encontrados durante e após o desenvolvimento. Para as funcionalidades do PTA, os defeitos foram encontrados 93% durante o desenvolvimento e somente 7% após o desenvolvimento, corresponde aos 15 defeitos. No módulo "PTR" e "PSA" foram encontrados durante o desenvolvimento 8 e 7 defeitos respectivamente. Sendo assim os percentual de defeitos encontrados durante o desenvolvimento, em 3/3 dos módulos é superior ao encontrado após o desenvolvimento.

A Figura 37 contempla os custos estimados de produção, retrabalho e totais dos módulos "PTA", "PSA" e "PTR". O custo total para o módulo "PTA", onde o tempo de desenvolvimento e correção foram de 15 dias, ficou em R\$2.288, sendo R\$ 291,20 resultantes de correções. Para o módulo "PSA" o custo total e de produção foi o mesmo R\$1.504, pois não sofreram influência de correções no módulo e o tempo de desenvolvimento durou 13 dias. Quanto ao módulo "PTR", assim como o "PSA", os custos de produção e total foram de R\$3.344, levando em consideração 22 dias para desenvolvimento. Sendo assim em 3/3 dos módulos o custo aproximado para correção é inferior ao custo de produção.

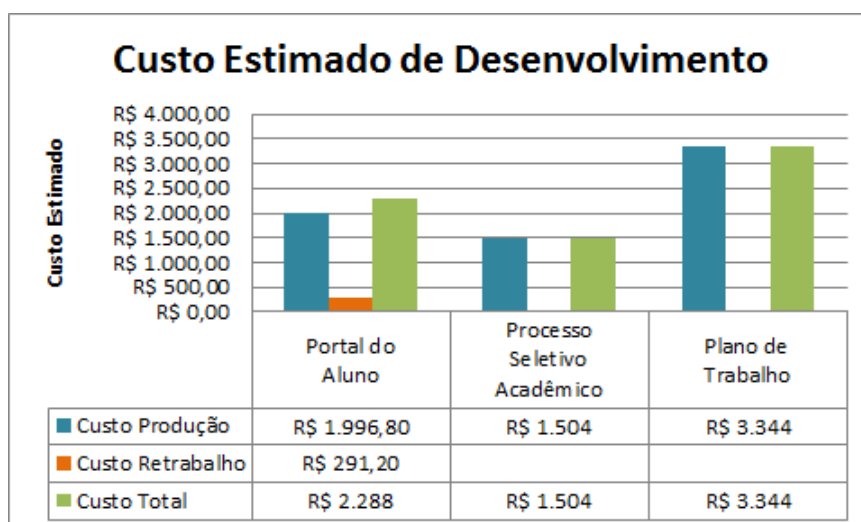
A distribuição do custo estimado, representado pela Figura 38, foi determinada pelo influência dos custo das correções que nesse caso só apareceu no módulo "PTA".

Figura 36 – PTA, PSA e PTR - Distribuição do Custo Estimado de Desenvolvimento .



Fonte:Elaborado Pelo Autor.

Figura 37 – PTA, PSA e PTR - Distribuição do Custo Estimado de Desenvolvimento .



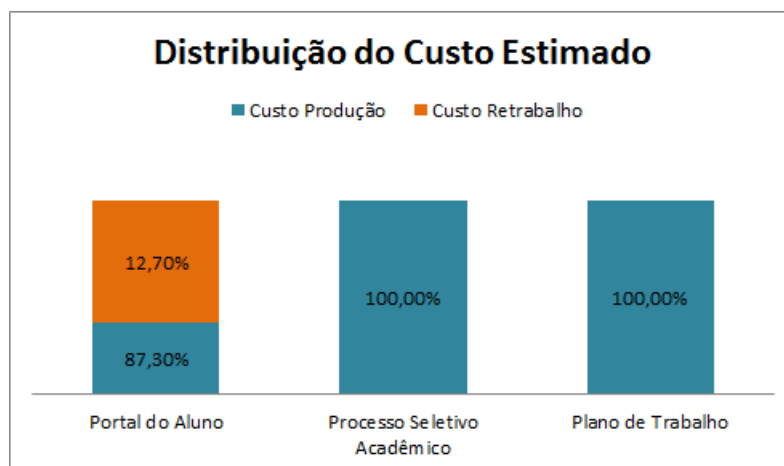
Fonte:Elaborado Pelo Autor.

Portanto o custo estimado de produção no módulo "PTA"corresponde a 87,3% do custo total do projeto e somente 12,7% do custo total é oriundo do retrabalho.

Com base nesses números, é identificado que no módulo "PTA", o custo de correção é o equivalente a 12,7% do total do projeto ou a 14,6% do custo de produção. Caso esse índice atingido pelo módulo "PTA"seja replicado ao módulo "PTP", haveria uma economia de R\$ 4195.42 ou aproximadamente 70% do valor gasto no módulo "PTP", que passaria a custar R\$1723.58 com o novo fator.

Se aplicarmos o índice de correção de 14,6% do custo de produção do "PTA"ao módulo "RU", haveria uma economia de R\$1.671,10 ou aproximadamente 32,88% do valor

Figura 38 – PTA, PSA e PTR- Distribuição do Custo Estimado de Desenvolvimento .



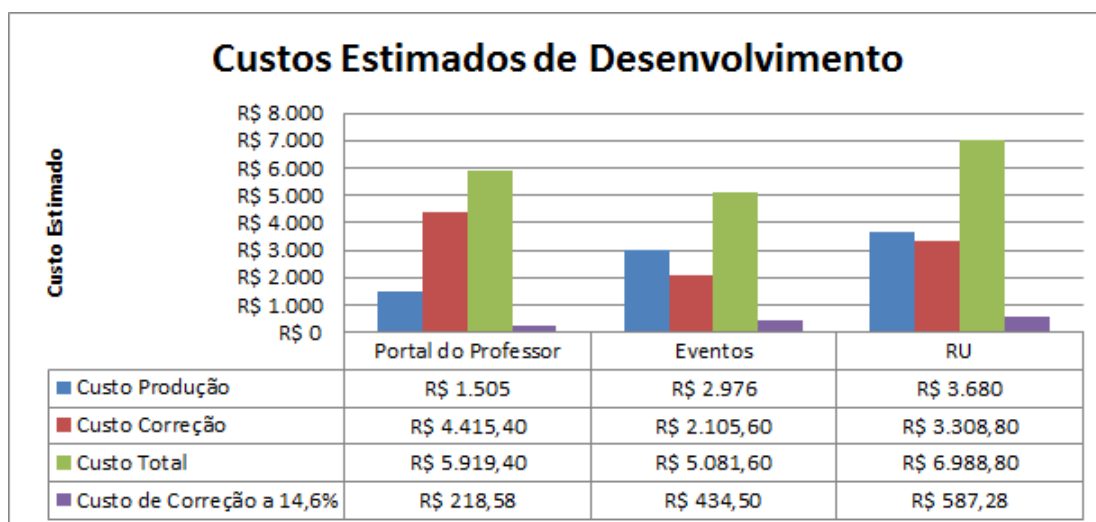
Fonte:Elaborado Pelo Autor.

gasto no módulo "RU", que passaria a custar R\$3.410,50.

Para as alterações no módulo "Eventos", aplicando esse mesmo índice de correção obtido no módulo "PTA" de 14,6%, a economia no módulo seria de R\$ 2.771,52 ou 39,6% do valor total, que seria alterado para R\$4.217,52.

A Figura 39 exibe os custos se aplicarmos o índice de 14,6% nos módulos PTP, EVT e RU.

Figura 39 – Custos de Desenvolvimento Considerando Correção a 14,6%.



Fonte:Elaborado Pelo Autor.

Essas estimativas foram feitas levando em consideração os desenvolvimentos acompanhados. O fato da aplicação do processo ocorrer em pacotes pequenos de desenvolvi-

mento, onde a complexidade lógica e gerencial é menor, reforçou a realização de um bom estudo de caso.

Como toda estimativa, acreditamos que o resultados possam variar se aplicados no desenvolvimento de um módulo completo, pois alteraria a complexidade de desenvolvimento. Porém ainda é possível identificar uma grande vantagem com o uso do novo processo.

7 Conclusão

Garantir a qualidade de sistemas continua sendo uma tarefa essencial e desafiadora para muitos Engenheiros de Software. Ainda que possua muitas definições, a qualidade de um produto como software é medida de forma superficial. Contemplar os requisitos solicitados pelo cliente e deixar o sistema livre de defeitos é o caminho mais adequado para definir se há realmente qualidade no processo e produto.

Considerando as dificuldades em garantir a qualidade de software, o objetivo desse trabalho é elaborar, a partir da revisão de literatura, dos trabalhos relacionados e da realidade de desenvolvimento do NTIC, um processo de Verificação e Validação que possa diminuir a incidência do retrabalho na CODEV.

A aplicação do processo de V&V, foi realizada em algumas funcionalidades desenvolvidas na CODEV. O estudo de caso demonstrou as atividades e as respostas obtidas nas funcionalidades. Com essa aplicação, pode-se observar que é possível agregar qualidade e características como confiabilidade e maior segurança aos software ali desenvolvidos.

De modo geral, foi possível observar a redução do retrabalho nos projetos em que foi utilizado o processo de V&V. Também foi possível estimar a redução dos custos de desenvolvimento quando há essa diminuição de custos com retrabalho.

O processo proposto para a CODEV, de Verificação e Validação, vem para auxiliar o desenvolvimento de software de forma a garantir qualidade, reduzir custos e a incidência do retrabalho. As informações geradas pelo processo, contribuem para afirmar os benefícios que um processo bem elaborado e executado pode trazer ao desenvolvimento de software.

7.1 Trabalhos Futuros

Com o término do presente trabalho, pode-se esperar como trabalhos futuros a manutenção e evolução do processo de V&V elaborado. Tendo em vista que o processo foi proposto levando em conta as necessidades atuais da CODEV é possível afirmar que as mesmas podem diferir em alguns anos, exigindo a evolução desse processo.

Outro passo interessante para a sequência desse trabalho é acompanhar a aplicação do processo no desenvolvimento de um módulo novo na CODEV, com o objetivo de analisar os resultados obtidos quando a complexidade de desenvolvimento e gerencial é maior.

Anexos

ANEXO A – Checklist para Inspeção de Código Elaborado pela CODEV

Checklist para verificação de código-fonte - NTIC

| | |
|-----------|--|
| Projeto: | |
| Analista: | |
| Data: | |

| Item | Realizado | Observações |
|--|-----------|-------------|
| 1. FORMATAÇÃO | | |
| 1.1 O código-fonte está identado e alinhado conforme o padrão? | | |
| 1.2 As linhas possuem menos de 80 caracteres (se possível)? | | |
| 1.3 A nomenclatura dos elementos está correta? (classes, métodos, funções, nomes de arquivos) | | |
| 1.4 A marcação xhtml está de acordo com as recomendações da W3C? | | |
| 2. COERÊNCIA | | |
| 2.1 As regras de negócio estão implementadas em controller/model e não na view? | | |
| 2.2 Existe algum código PHP na view que possa ser movido para a controller afim de tornar a view mais limpa? | | |
| 2.3 Existe algum método/função que pode ser movido para outro arquivo tendo em vista a reutilização? | | |
| 2.4 Existem trechos de código repetidos que possam ser convergidos numa função? | | |
| 2.5 Nos formulários, os campos obrigatórios estão sendo validados corretamente? | | |
| 2.6 Nos formulários, os campos não obrigatórios estão sendo tratados corretamente? | | |
| 2.7 Na abertura das views, existe algum código "pesado" executando na controller que possa ser melhorado? exemplo: muitas consultas SQL ou consultas SQL mal formuladas ou trazendo mais resultados do que o necessário? | | |
| 2.8 O charset de arquivos e do banco de dados está de acordo com a programação/exibição na tela? | | |

| 3. BOAS PRÁTICAS | | |
|---|--|--|
| 3.1 Existe algum trecho de código com que possa ser reformulado tendo em vista a legibilidade? (com muitos if's aninhados, laços e operadores que possam ser simplificados) | | |
| 3.2 Existe algum trecho de código em que possa ser usado try/catch tendo em vista a legibilidade e eficiência? (exemplo: validações de edição e exclusão) | | |
| 3.3 Todas as leituras de variáveis estão protegidas contra "undefined"? | | |
| 3.4 O código está documentado/comentado adequadamente? | | |
| 4. SEGURANÇA | | |
| 4.1 Formulários públicos que salvam dados em banco e/ou enviam e-mail estão protegidos com captcha? | | |
| 4.2 Existem recursos restritos acessíveis diretamente via URL? | | |

Apêndices

APÊNDICE A – Template para Casos de Teste

APÊNDICE B – 1529 Atividade 1 - Saída da Inspeção de Requisitos

CENÁRIOS

GURI-PTA-CDU-001-Visualizar Plano de Ensino (Aluno)

| | |
|----------------------------|---|
| Descrição | Este caso de uso descreve a visualização do plano de ensino de uma disciplina por parte de um aluno. |
| Ator(es) | Aluno |
| Pré-Condição | - Existir usuário logado. - Ser aluno. - As disciplinas em que o plano de ensino não estiver finalizado deverão aparecer para o aluno, mas não será possível visualizar o plano. - A visualização será dada em PDF. |
| Fluxo Principal | <ul style="list-style-type: none">• P1. O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino• P2. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso.• P3. Aluno imprime o PDF do plano de ensino. |
| Fluxos Alternativos | <ul style="list-style-type: none">• A01. Aluno pesquisa plano de ensino:<ul style="list-style-type: none">○ A01. P1. Aluno clica no ícone busca○ A01. P2. Aluno efetua pesquisa por ano, semestre curso e/ou disciplina. |
| Exceções | 1. Visualização do plano somente estará disponível se professor tiver finalizado o plano de ensino. |

GURI-PTA-CDU-002-Visualizar Diário de Classe (Aluno)

| | |
|----------------------------|--|
| Descrição | Este caso de uso descreve a visualização do diário de classe de uma disciplina por parte de um aluno. |
| Ator(es) | Aluno |
| Pré-Condição | - Existir usuário logado. - Ser aluno. - A visualização será dada em HTML. |
| Fluxo Principal | <ul style="list-style-type: none">• P1. O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Diário de Classe• P2. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso.• P3. Aluno seleciona a disciplina.• P4. Aluno visualiza o Diário de Classe, contendo todas as aulas finalizadas, com data, conteúdo ministrado e registro de falta do discente na aula (em HTML). |
| Fluxos Alternativos | <ul style="list-style-type: none">• A01. Aluno pesquisa Diário de Classe:<ul style="list-style-type: none">○ A01. P1. Aluno clica no ícone busca○ A01. P2. Aluno efetua pesquisa por ano, semestre curso e/ou disciplina. |
| Exceções | 1. Somente aparecerão as aulas do diário de classe em que a aula tenha sido finalizada pelo professor. As demais não deverão aparecer para o aluno. 2. Registros de faltas de outros discentes não deverão estar disponíveis. |

APÊNDICE C – 1529 Atividade 2 e 7 -
Saída da Criação dos Casos de Teste e Testes
de Aceitação Executados

| GURI Módulo Portal do Aluno | | | | | | | | |
|-----------------------------|------------------------------|------------------|---|--|---|--|---|--|
| Código | Projeto | ID Caso de Uso | Caso de Uso | Pré-condições | Entrada(s) | Resultado Esperado | Resultado Obtido | Observações |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-001 | Visualizar Plano de Ensino (Aluno) | O usuário "aluno" está logado e o professor deve ter finalizado o plano de ensino. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. O aluno clica no botão visualizar. | O sistema faz o download do plano de ensino em formato pdf. | O sistema está fazendo o download do plano de ensino corretamente. Formato pdf ok. | O sistema está fazendo o download do plano de ensino corretamente. Formato pdf ok. |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-001 | Visualizar Plano de Ensino (Aluno) | O usuário "aluno" está logado e o professor deve ter finalizado o plano de ensino. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. O aluno clica no botão visualizar e o sistema exibe o pdf. | O sistema faz o download do pdf contendo as informações referentes ao plano de ensino da disciplina desejada. | O sistema está retornando as informações corretas no plano de ensino. São apresentadas os Dados de Identificação, Ementa, Objetivo Geral, Objetivos Específicos, Metodologia, Avaliação, Atividades de recuperação, Cronograma, Atendimento aos acadêmicos, Ações Interdisciplinares, Outras Ações, Bibliografia básica e complementar. | Não está respeitando a seguinte pré-condição: O plano de ensino precisa ter sido finalizado pelo professor. |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-001 | Visualizar Plano de Ensino (Aluno) | O usuário "aluno" está logado. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno clica no ícone busca seleciona ano. | O sistema retorna os planos de ensino correspondentes ao ano pesquisado. | O sistema está exibindo a busca correta pelo ano. | |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-001 | Visualizar Plano de Ensino (Aluno) | O usuário "aluno" está logado. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno clica no ícone busca seleciona Semestre. | O sistema retorna os planos de ensino correspondentes ao semestre pesquisado. | O sistema não efetua a busca pelo semestre. FALHOU NO TESTE | Não conseguiu efetuar a busca por semestre. Utilize os seguintes termos: "1,2", "1", "2", "um", "dois", "2", "2 semestre". |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-001 | Visualizar Plano de Ensino por Disciplina | O usuário "aluno" está logado. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno clica no ícone busca seleciona disciplina. | O sistema retorna os planos de ensino correspondentes a disciplina pesquisada. | O sistema está efetuando a busca e correta pela disciplina. | |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-001 | Visualizar Plano de Ensino (Aluno) | O usuário "aluno" está logado. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno clica no ícone busca seleciona ID Plano. | O sistema retorna os planos de ensino correspondentes ao ID Plano pesquisado. | O sistema está efetuando a busca correta pelo ID. | |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-001 | Visualizar Plano de Ensino (Aluno) | O usuário "aluno" está logado. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno clica no ícone busca seleciona ID Plano. | O sistema retorna os planos de ensino correspondentes ao ID Plano pesquisado. | O sistema está efetuando a busca correta pelo ID. | |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-001 | Visualizar Plano de Ensino (Aluno) | O usuário "aluno" está logado e o professor deve ter finalizado o plano de ensino. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Plano de Ensino Aluno. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. O aluno clica no botão visualizar. | O sistema retorna somente o download de planos de ensino finalizados pelo docente no Portal do Professor. | O sistema está permitindo o download de planos de ensino não finalizados. FALHOU NO TESTE | Não está respeitando a seguinte pré-condição: O plano de ensino precisa ter sido finalizado pelo professor. |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-002 | Visualizar Diário de Classe (Aluno) | O usuário "aluno" está logado. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Diário de Classe. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno seleciona a disciplina. | O sistema exibe o diário de classe em formato html. | O sistema está retornando o diário de classe correspondente a disciplina solicitada. Formato html ok | |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-002 | Visualizar Diário de Classe (Aluno) | O usuário "aluno" está logado e o professor deve ter finalizado a aula no sistema. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Diário de Classe. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno seleciona a disciplina. | O sistema retorna as seguintes informações quanto ao diário de classe: todas as aulas finalizadas, com data, conteúdo ministrado e registro de fala do discente na aula. | O sistema está retornando as informações correlatas quanto ao diário de classe. | |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-002 | Visualizar Diário de Classe (Aluno) | O usuário "aluno" está logado e o professor deve ter finalizado a aula no sistema. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Diário de Classe. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno seleciona a disciplina. | O sistema exibe somente as aulas que foram finalizadas pelo professor. | As informações retornadas quanto as aulas estão corretas. | Podem ser interessante simular (testar) a inclusão de aula finalizada e não finalizada por um professor e conferir no diário de classe do aluno. |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-002 | Visualizar Diário de Classe (Aluno) | O usuário "aluno" está logado e o professor deve ter finalizado a aula no sistema. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Diário de Classe. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. Aluno seleciona a disciplina. | O sistema retorna as disciplinas em que o aluno está matriculado e somente as que os professores tenham disponibilizado o diário de classe. | O sistema está retornando somente a frequência correspondente ao aluno que está visualizando o diário de classe. | |
| 1529 | GURI - PTA - Portal do Aluno | GURI-PTA-CDU-002 | Visualizar Diário de Classe (Aluno) | O usuário "aluno" está logado. | O Aluno entra na funcionalidade Acadêmico -> Portal do Aluno -> Diário de Classe. O Aluno visualiza todas as disciplinas separadas por ano, semestre e curso. | O sistema retorna as informações correlatas quanto ao diário de classe. | O sistema está retornando as informações correlatas quanto ao diário de classe. | OK |

APÊNDICE D – 1529 Atividade 5 - Saída da Inspeção de Código

Checklist para verificação de código-fonte - NTIC

| Projeto: | PTA - Portal do Aluno (caso #1529) - Plano de Ensino | | |
|--|---|------------------------|--|
| Analista: | | | |
| Data da Revisão: | | | |
| Estagiária: | | | |
| Item | Observação | Realizado | Retorno |
| 1. FORMATAÇÃO | | | |
| 1.1 O código-fonte está indentado e alinhado conforme o padrão? | | | |
| 1.2 As linhas possuem menos de 80 caracteres (se possível)? | | Ok | |
| 1.3 A nomenclatura dos elementos está correta? (classes, métodos, funções, nomes de arquivos) | Procurar utilizar 3 letras para formar segmento de nomes de função (ex. getCronogramaAulaNPresencial -> getCronogramaAulaNaoPresencial). | Verificado e corrigido | Favor detalhar o que foi realizado. FOI REVISADO AS FUNÇÕES QUE O NOME NÃO ESTAVAM NO PADRÃO |
| 1.4 A marcação xhtml está de acordo com as recomendações da W3C? | | Ok | |
| 2. COERÊNCIA | | | |
| 2.1 As regras de negócio estão implementadas em controller/model e não na view? | | Ok | |
| 2.2 Existe algum código PHP na view que possa ser movido para a controller afim de tornar a view mais limpa? | | Ok | |
| 2.3 Existe algum método/função que pode ser movido para outro arquivo tendo em vista a reutilização? | Retirar funções não utilizadas que possam ter sido copiadas do esqueleto | Ok | |
| 2.4 Existem trechos de código repetidos que possam ser convergidos numa função? | | | |
| 2.5 Nos formulários, os campos obrigatórios estão sendo validados corretamente? | | | |
| 2.6 Nos formulários, os campos não obrigatórios estão sendo tratados corretamente? | | Ok | |
| 2.7 Na abertura das views, existe algum código "pesado" executando na controller que possa ser melhorado? exemplo: muitas consultas SQL ou consultas SQL mal formuladas ou trazendo mais resultados do que o necessário? | Verificar lentidão de acesso a dados | Verificado | As consultas de listagem (search) estão lentas ainda. Neste momento em Testes não estão retornando dado algum, diferente do último teste realizado. Procurar não usar variáveis de sessão na model e sim parâmetros na chamada da função. Tratar estes valores na controller! FOI RETIRADA AS VARIÁVEIS DE SESSÃO DA MODEL |
| 2.8 O charset de arquivos e do banco de dados está de acordo com a programação/exibição na tela? | | Ok | |
| 3. BOAS PRÁTICAS | | | |
| 3.1 Existe algum trecho de código com que possa ser reformulado tendo em vista a legibilidade? (com muitos if's aninhados, laços e operadores que possam ser simplificados) | Revisar ifs aninhados e chaves (ex. linha 120 da controller). - Procurar deixar espaços em branco entre blocos para melhorar a legibilidade (entre atribuições e blocos de if, por exemplo) | Verificado e corrigido | Ainda há blocos de código "colados" que dificultam a legibilidade Revisado CORRIGIDO |
| 3.2 Existe algum trecho de código em que possa ser usado try/catch tendo em vista a legibilidade e eficiência? (exemplo: validações de edição e exclusão) | Utilizar switch na linha 142 da controller | | |
| 3.3 Todas as leituras de variáveis estão protegidas contra "undefined"? | | Ok | |
| 3.4 O código está documentado/comentado adequadamente? | Revisar comentários de funções e cabeçalhos de arquivo (comentários copiados de outro módulo) | Verificado e corrigido | Persiste a questão de comentários errados (ex. Nome de um desenvolvedor com o e-mail/nome de outro) CORRIGIDO |
| 4. SEGURANÇA | | | |
| 4.1 Formulários públicos que salvam dados em banco e/ou enviam e-mail estão protegidos com captcha? | Não se aplica - Diminuir comprimento da URL, retirando informações que possam permitir injeção de SQL (Segurança!!) | | |
| 4.2 Existem recursos restritos acessíveis diretamente via URL? | | Verificado | Impossível testar. Não há dados nas consultas de listagem. |
| Sugestões | | | |
| Plano de Ensino | | | |
| - Nomear planos de ensino como PlanoEnsino-ALXXXX (com o código da disciplina) | | | |
| - Tempo para geração do PDF muito ALTO | | | |
| - Diminuir comprimento da URL, retirando informações que possam permitir injeção de SQL (Segurança!!) | | | |
| - Evitar deixar linhas em branco ao final dos arquivos (compatibilidade) | | | |

Checklist para verificação de código-fonte - NTIC

| Projeto: | PTA - Portal do Aluno (caso #1529) - Diário de Classe | | |
|--|---|--|---------|
| Analista: | | | |
| Data da Revisão: | | | |
| Estagiária: | | | |
| Item | Observação | Realizado | Retorno |
| 1. FORMATAÇÃO | | | |
| 1.1 O código-fonte está identado e alinhado conforme o padrão? | | Ok | |
| 1.2 As linhas possuem menos de 80 caracteres (se possível)? | | Ok | |
| 1.3 A nomenclatura dos elementos está correta? (classes, métodos, funções, nomes de arquivos) | | Ok | |
| 1.4 A marcação xhtml está de acordo com as recomendações da W3C? | Verificar visualização do diário de classe e marcação de ícones. Há ícones não enviados e com ALT trocado. Verificar se não podem ser utilizados outros ícones na marcação das presenças que não sejam confundidos com o uso do ícone em outras telas. Retirar ou renomear elementos <label> desnecessários ou nomeados erroneamente. | Ok para listagem de disciplinas. Verificar depois da correção com plugin validador ou diretamente no site w3c. | |
| 2. COERÊNCIA | | | |
| 2.1 As regras de negócio estão implementadas em controller/model e não na view? | | Ok | |
| 2.2 Existe algum código PHP na view que possa ser movido para a controller afim de tornar a view mais limpa? | | Ok | |
| 2.3 Existe algum método/função que pode ser movido para outro arquivo tendo em vista a reutilização? | Retirar funções não utilizadas que possam ter sido copiadas do esqueleto | | |
| 2.4 Existem trechos de código repetidos que possam ser convergidos numa função? | | Ok | |
| 2.5 Nos formulários, os campos obrigatórios estão sendo validados corretamente? | Não se aplica | Ok | |
| 2.6 Nos formulários, os campos não obrigatórios estão sendo tratados corretamente? | Não se aplica | Ok | |
| 2.7 Na abertura das views, existe algum código "pesado" executando na controller que possa ser melhorado? exemplo: muitas consultas SQL ou consultas SQL mal formuladas ou trazendo mais resultados do que o necessário? | | Ok | |
| 2.8 O charset de arquivos e do banco de dados está de acordo com a programação/exibição na tela? | | Ok | |
| 3. BOAS PRÁTICAS | | | |
| 3.1 Existe algum trecho de código com que possa ser reformulado tendo em vista a legibilidade? (com muitos ifs aninhados, laços e operadores que possam ser simplificados) | Revisar espaços entre blocos de atribuição e controle | | |
| 3.2 Existe algum trecho de código em que possa ser usado try/catch tendo em vista a legibilidade e eficiência? (exemplo: validações de edição e exclusão) | Erro de variável undefined em http://localhost/erp/trunk/src/pta/diario/visualizar/13144 | | |
| 3.3 Todas as leituras de variáveis estão protegidas contra "undefined"? | Revisar comentários de funções e cabeçalhos de arquivo (comentários copiados de outro módulo) | | |
| 3.4 O código está documentado/comentado adequadamente? | | | |
| 4. SEGURANÇA | | | |
| 4.1 Formulários públicos que salvam dados em banco e/ou enviam e-mail estão protegidos com captcha? | Não se aplica | Ok | |
| 4.2 Existem recursos restritos acessíveis diretamente via URL? | | Ok | |
| Sugestões | | | |
| Diário de Classe | | | |
| - Retirar funções excedentes e não utilizadas. | | | |
| - Retirar funções de teste do código | | | |
| - Revisar erros de digitação na nomenclatura de arquivos. | | | |
| - Retirar linhas em branco em excesso | | | |

APÊNDICE E – 1635 Atividade 1 - Saída da Inspeção de Requisitos

GURI-PSA-CDU-001-Habilitar Anexos de Arquivos Durante Inscrição

| | |
|-----------------------------|--|
| Descrição/Requisitos | Criar filtro para habilitar envio de anexos pelo módulo de Processos Seletivos Acadêmicos. |
| Ator (es) | Gerenciador de Processos Seletivos |
| Fluxo Principal | <p>P1. Usuário acessa Acadêmico -> Processo Seletivo -> Processos Seletivos;</p> <p>P2. Sistema exibe processos seletivos cadastrados;</p> <p>P3. Usuário clica em editar, na linha correspondente ao processo seletivo desejado;</p> <p>P4. Usuário preenche todos os campos obrigatórios (inclusive novo campo que habilita ou não envio de anexos pelos inscritos via PSA) e clica no botão Salvar.</p> |
| Exceções | Somente será possível anexar arquivos do tipo PDF. |

GURI-PSA-CDU-002-Anexar Arquivos

| | |
|-----------------------------|--|
| Descrição/Requisitos | Anexar arquivos durante inscrição de processos seletivos acadêmicos |
| Ator (es) | Candidato |
| Pré-Condição | 1. Envio de arquivos habilitado via configuração do processo seletivo (CDU-001). |
| Fluxo Principal | <p>P1. Usuário acessa Acadêmico -> Processo Seletivo -> Nova Inscrição;</p> <p>P2. Sistema exibe processos seletivos cadastrados;</p> <p>P3. Usuário clica no processo seletivo desejado;</p> <p>P4. Sistema exibe acesso à área do candidato;</p> <p>P5. Usuário preenche todos os dados;</p> <p>P6. Se configuração estiver habilitada para envio de arquivos, sistema</p> |
| | <p>exibe modal para emissão de arquivos no formato PDF;</p> <p>P7. Usuário envia arquivos desejados e salva;</p> <p>P8. Sistema exibe área do candidato.</p> |
| Exceções | <ol style="list-style-type: none"> 1. Somente será possível adicionar arquivos do tipo PDF. 2. Usuário poderá adicionar quantos arquivos quiser. |

GURI-PSA-CDU-003-Visualizar Arquivos Enviados

| | |
|-----------------------------|--|
| Descrição/Requisitos | Visualização de Arquivos emitidos via inscrição no módulo PSA. |
| Ator (es) | Candidato |
| Pré-Condição | 1. Haver inscrição ativa do candidato. |
| Fluxo Principal | P1. Usuário acessa Acadêmico -> Processo Seletivo -> Área do Candidato; P2. Sistema exibe inscrições do candidato; P3. Usuário clica em Visualizar Arquivos; P4. Módulo gera Thumbnails contendo miniaturas de imagens das primeiras páginas de cada arquivo da inscrição, sendo um dos thumbnails, em destaque, contendo todos os processos no mesmo arquivo em PDF; P5. Usuário clica em um dos arquivos; P6. Módulo de inscrições utiliza o script PDF.JS para mostrar o PDF na tela não necessitando que o usuário tenha um visualizador de PDF instalado em sua máquina. |
| Exceções | 1. Somente será possível visualizar arquivos do tipo PDF. |

GURI-PSA-CDU-004-Visualizar Arquivos Enviados (Administração)

| | |
|-----------------------------|--|
| Descrição/Requisitos | Visualização de Arquivos emitidos via inscrição no módulo PSA. |
| Ator (es) | Gerenciador de Processos Seletivos |
| Pré-Condição | 1. Haver inscrições ativas de candidatos. |
| Fluxo Principal | P1. Usuário acessa Acadêmico -> Processo Seletivo -> Inscrições em Processos Seletivos; P2. Sistema exibe inscrições; P3. Usuário clica Visualizar Arquivos; P4. Módulo gera Thumbnails contendo miniaturas de imagens das primeiras páginas de cada arquivo da inscrição, sendo um dos thumbnails, em destaque, contendo todos os processos no mesmo arquivo em PDF; P5. Usuário clica em um dos arquivos; P6. Módulo de inscrições utiliza o script PDF.JS para mostrar o PDF na tela não necessitando que o usuário tenha um visualizador de PDF instalado em sua máquina. |
| Exceções | 1. Somente será possível visualizar arquivos do tipo PDF. |

APÊNDICE F – 1635 Atividade 2 e 7 -
Criação dos Casos de Teste e Testes de
Aceitação Executados

| GURI Módulo Processo Seletivo Acadêmico | | | | | | | | |
|---|------------------|--|--|--|--|------------------------------------|--|-------------|
| Código Projeto | ID Caso de Uso | Caso de Uso | Objetivo do teste | Pré-condições | Entrada(s) | Resultado Esperado | Resultado Obtido | Observações |
| 000.1635 GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-001 | Manter Processos Seletivos | Habilitar envio de anexos pelo PSA | O usuário deve estar logado como "Gerenciador" de PSA | Usuário deve acessar o menu Acadêmico -> Processo Seletivo -> Processos Seletivos, clicar em editar na linha correspondente ao processo seletivo desejado, preencher todos os campos obrigatórios, habilitando a opção de envio de anexos selecionando SIM para upload de arquivos, e clicar no botão "Salvar". | Caracteres alfabéticos e numéricos | O sistema deve salvar as alterações feitas, habilitar a opção de "Upload de arquivos" e exibir um feedback ao usuário. | OK |
| 000.1635 GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-001 | Manter Processos Seletivos | Não Habilitar o envio de anexos pelo PSA | Não Habilitar o envio de anexos pelo PSA | Usuário deve acessar o menu Acadêmico -> Processo Seletivo -> Processos Seletivos, clicar em editar na linha correspondente ao processo seletivo desejado, preencher todos os campos obrigatórios, não habilitando a opção de envio de anexos selecionando NÃO para upload de arquivos, e clicar no botão "Salvar". | Caracteres alfabéticos e numéricos | O sistema deve salvar as alterações feitas, não habilitar a opção de "Upload de arquivos" e exibir um feedback ao usuário. (mensagem) | OK |
| 000.1635 GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-001 | Manter Processos Seletivos | Cancelar a | O usuário deve estar logado como "Gerenciador" de PSA | Usuário deve acessar o menu Acadêmico -> Processo Seletivo -> Processos Seletivos, clicar em editar na linha correspondente ao processo seletivo desejado, preencher todos os campos obrigatórios, habilitando a opção de envio de anexos selecionando SIM para upload de arquivos, e clicar no botão "Cancelar". | Caracteres alfabéticos e numéricos | O sistema deve cancelar as alterações, não habilitar a opção de "Upload de Arquivos" e retornar a página anterior. (deveria exibir uma mensagem de feedback) | OK |
| 000.1635 GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-002 | Anexar Arquivos | Anexar arquivo durante inscrição de PSA | Anexar arquivo durante inscrição de PSA | Usuário acessa o menu Acadêmico -> Processo Seletivo -> Nova Inscrição. O Usuário clica no processo seletivo desejado e preenche todos os dados solicitados. O usuário clica na opção para fazer o upload de arquivos, clica no botão "Procurar", seleciona um arquivo PDF, clica no botão "Anexar Arquivos" e clica no botão "Salvar". | Arquivo PDF | O sistema deve salvar o arquivo anexado juntamente com inscrição do candidato, exibir um feedback ao usuário (mensagem) e redirecionar a área do candidato. | OK |
| 000.1635 GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-002 | Anexar Arquivos | Anexar arquivo em formato diferente do permitido | Anexar arquivo em formato diferente do permitido | Usuário acessa o menu Acadêmico -> Processo Seletivo -> Nova Inscrição. O Usuário clica no processo seletivo desejado e preenche todos os dados solicitados. O usuário clica na opção para fazer o upload de arquivos, clica no botão "Procurar", seleciona um arquivo e clica no botão "Anexar Arquivos". | Arquivo (doc, odt, png...) | O sistema não deve salvar o arquivo anexado e deve exibir uma mensagem informando que só podem ser anexados arquivos pdf. | OK |
| 000.1635 GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-002 | Anexar Arquivos | Anexar vários arquivos durante a inscrição do PSA | Anexar vários arquivos durante a inscrição do PSA | Usuário acessa o menu Acadêmico -> Processo Seletivo -> Nova Inscrição. O Usuário clica no processo seletivo desejado e preenche todos os dados solicitados. O usuário clica na opção para fazer o upload de arquivos, clica no botão "Procurar", seleciona um arquivo PDF, clica no botão "Adicionar Arquivo", repete a operação até adicionar todos os arquivos desejados, clica no botão "Anexar Arquivos" e clica no botão "Salvar". | Arquivo PDF | O sistema deve salvar o arquivo anexado juntamente com inscrição do candidato, exibir um feedback ao usuário (mensagem) e redirecionar a área do candidato. | Não |
| 000.1635 GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-002 | Anexar Arquivos | Não anexar arquivos durante a inscrição (objetivo é verificar se anexar os arquivos está respeitando o caráter opcional) | Não anexar arquivos durante a inscrição (objetivo é verificar se anexar os arquivos está respeitando o caráter opcional) | Usuário acessa o menu Acadêmico -> Processo Seletivo -> Nova Inscrição. O Usuário clica no processo seletivo desejado e preenche todos os dados solicitados e clica no botão "Anexar Arquivos". | Arquivo PDF | O sistema deve salvar a inscrição do candidato, exibir um feedback ao usuário (mensagem) e redirecionar a área do candidato. | OK |
| 000.1635 GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-003 | Visualizar Arquivos Enviados | Visualizar arquivos emitidos via inscrição no módulo PSA | Visualizar arquivos emitidos via inscrição no módulo PSA | Usuário acessa o menu Acadêmico -> Processo Seletivo -> Área do Candidato. O Usuário clica em "Visualizar Arquivos", na linha correspondente ao arquivo desejado. | | O sistema exibe o arquivo em formato pdf na tela. | OK |
| 000.1635 GURI – PSA – Processo Seletivo Acadêmico | GURI-PSA-CDU-004 | Visualizar Arquivos Enviados (Administração) | Visualizar arquivos emitidos via inscrição no módulo PSA | Visualizar arquivos emitidos via inscrição no módulo PSA | Usuário acessa o menu Acadêmico -> Processos Seletivos. O Usuário seleciona "Visualizar Arquivos" na coluna Ação, correspondente ao candidato que deseja visualizar. | | O sistema exibe o arquivo em formato pdf na tela. | OK |

APÊNDICE G – 1635 Atividade 5 - Inspeção de Código

Checklist para verificação de código-fonte - NTIC

| Projeto: | Processo Seletivo Acadêmico - Caso #1635 | | |
|--|--|-----------|--------|
| Analista: | | | |
| Data da Revisão: | | | |
| | | | |
| Item | Observação | Realizado | Retomo |
| 1. FORMATAÇÃO | | | |
| 1.1 O código-fonte está indentado e alinhado conforme o padrão? | Rever o alinhamento do código e tabulações | | |
| 1.2 As linhas possuem menos de 80 caracteres (se possível)? | | Ok | |
| 1.3 A nomenclatura dos elementos está correta? (classes, métodos, funções, nomes de arquivos) | | Ok | |
| 1.4 A marcação xhtml está de acordo com as recomendações da W3C? | Não se aplica - checklist apenas de código | | |
| 2. COERÊNCIA | | | |
| 2.1 As regras de negócio estão implementadas em controller/model e não na view? | | Ok | |
| 2.2 Existe algum código PHP na view que possa ser movido para a controller afim de tornar a view mais limpa? | | Ok | |
| 2.3 Existe algum método/função que pode ser movido para outro arquivo tendo em vista a reutilização? | | Ok | |
| 2.4 Existem trechos de código repetidos que possam ser convergidos numa função? | | Ok | |
| 2.5 Nos formulários, os campos obrigatórios estão sendo validados corretamente? | | Ok | |
| 2.6 Nos formulários, os campos não obrigatórios estão sendo tratados corretamente? | | Ok | |
| 2.7 Na abertura das views, existe algum código "pesado" executando na controller que possa ser melhorado? exemplo: muitas consultas SQL ou consultas SQL mal formuladas ou trazendo mais resultados do que o necessário? | | Ok | |
| 2.8 O charset de arquivos e do banco de dados está de acordo com a programação/exibição na tela? | | Ok | |
| 3. BOAS PRÁTICAS | | | |
| 3.1 Existe algum trecho de código com que possa ser reformulado tendo em vista a legibilidade? (com muitos ifs aninhados, laços e operadores que possam ser simplificados) | Melhorar alinhamento e tabulação, espaços entre blocos | | |
| 3.2 Existe algum trecho de código em que possa ser usado try/catch tendo em vista a legibilidade e eficiência? (exemplo: validações de edição e exclusão) | | Ok | |
| 3.3 Todas as leituras de variáveis estão protegidas contra "undefined"? | | Ok | |
| 3.4 O código está documentado/comentado adequadamente? | Faltam comentários em diversas funções | Não | |
| 4. SEGURANÇA | | | |
| 4.1 Formulários públicos que salvam dados em banco e/ou enviam e-mail estão protegidos com captcha? | | Ok | |
| 4.2 Existem recursos restritos acessíveis diretamente via URL? | | Ok | |
| | | | |
| Recomendado: revisão geral do módulo | | | |

—

APÊNDICE H – 1387 Atividade 1 - Saída da Inspeção de Requisitos

GURI-PTR-CDU-002-Manter Plano de Trabalho
(inserir, editar, copiar e enviar p/chefia)(servidor)

| | |
|----------------------------|---|
| Descrição | Este caso de uso descreve as seguintes operações: a. Inserção de dados: O servidor confere dados profissionais e cadastrará um plano de trabalho com informações de áreas de atividade e um tempo previsto para cada uma; b. Editar Plano: Com base nas informações das observações da |
| | chefia o servidor edita as atividades. A atividade deve ficar esmaecida e uma nova deve substituí-la. Ambas devem ser mostradas. TODAS as alterações devem ser ADICIONADAS. NUNCA substituídas; c. Clonar Plano: O servidor clona a última versão de cada atividade inserida. depois pode editá-las antes de enviar para a chefia; d. Enviar para chefia: O servidor envia seu plano para sua chefia avaliar. |
| Ator (es) | Servidor |
| Pré-Condção | Existir usuário logado e ser servidor Existir Descrição de Atividades cadastradas |
| Fluxo Principal | <ul style="list-style-type: none"> • P1. O Servidor entra na funcionalidade; • P2. O sistema apresenta as opções: Inserção, Edição (listagem), Clonagem (listagem), Enviar (listagem); A listagem é de planos com situação 1 ou 3; Situação 5 ou 6 devem aparecer apenas para visualização. Ver TELA 01; • P3. O Servidor seleciona inserção de dados; • P4. O sistema busca as informações do Servidor (Logado no GURI) no SIE (Nome, siape, cargo, local de exercício, Carga Horário) preenche o campo Ano referência com o ano corrente; Ver TELA 02; • P5. O Servidor confere o <u>nome</u>, <u>Siape</u>, <u>Cargo</u>, <u>Carga Horário</u> e <u>ano corrente</u>. Pode alterar chefia e local de exercício; seleciona uma das áreas de atividades (categorias) (Administrativas/Técnicas, Gestão/Representação, Extensão, Pesquisa e Ensino) para inserir N atividades) e um tempo previsto geral por categoria; Salva as informações; • P6. O Sistema seta a situação "1" caso e armazena as informações. |
| Fluxos Alternativos | <ul style="list-style-type: none"> • A01. P3. O Servidor seleciona "Editar Plano" na lista de registros localizados (situação 1 ou 3): <ul style="list-style-type: none"> ◦ A01. P3. a. O sistema busca e mostra o registro; ◦ A01. P3. b. O Servidor altera as informações necessárias e vai em salvar; ◦ A01. P3. c. O sistema salva as informações e retorna ao passo P2 do fluxo principal; • A02. P3. O Servidor seleciona "Clonar Plano": <ul style="list-style-type: none"> ◦ A02. P3. a. O sistema busca o registro do ano anterior, com a última versão de cada atividade, e apresenta na tela de formulário de novo registro; Ver TELA 02; ◦ A02. P3. b. O Servidor altera as informações necessárias; ◦ A02. P3. c. O Servidor salva as informações; ◦ A02. P3. d. O sistema coloca como situação "1" e retorna ao passo P2 do fluxo principal. • A03. P3. O Servidor seleciona "Enviar Plano": <ul style="list-style-type: none"> ◦ A03. P3. a. O sistema coloca como situação "2" e retorna ao passo P2 do fluxo principal. |
| Exceções | |

GURI-PTR-CDU-003-Visualizar Plano de Trabalho

| | |
|----------------------------|--|
| Descrição | Descreve o fluxo para visualizar os planos de trabalho. Qualquer Servidor do sistema poderá visualizar o plano, mas não podem ver as observações. |
| Ator (es) | Servidor |
| Pré-Condição | Servidor logado; Plano de trabalho inserido.. |
| Fluxo Principal | <ul style="list-style-type: none"> • P1. O Servidor entra na funcionalidade; • P2. O sistema apresenta a lista com planos cadastrados com situação 5 apenas do ano anterior ao corrente; • P3. O Servidor seleciona o plano; • P4. O sistema busca informações e apresenta para o Servidor com a CH por categoria (não apresenta Observações). |
| Fluxos Alternativos | |
| Exceções | |

GURI-PTR-CDU-004-Inserir Observações no Plano de Trabalho (chefia imediata)

| | |
|----------------------------|--|
| Descrição | A chefia somente adiciona informações ao plano de trabalho do subordinado. Todas as observações devem ficar salvas e não podem ser apagadas. A chefia somente devolve ao servidor uma vez, situação NÃO REFERENCIADA (pendente de ajustes). Na segunda passada pela chefia fica na situação REFERENCIADA (aprovada). O plano ficará aberto até o final do ano/referência para acréscimo de informações. |
| Ator (es) | Servidor (chefe) |
| Pré-Condição | Servidor logado; Plano de trabalho inserido com situação 2 ou 4; |
| Fluxo Principal | <ul style="list-style-type: none"> • P1. O Servidor entra na funcionalidade; • P2. O sistema verifica se o servidor logado é chefe de outro servidor e lista os planos na situação “2” ou “4”; Ver TELA 03; • P3. O Servidor seleciona o plano “2”; • P4. O sistema busca as informações e apresenta; • P5. O Servidor confere e aprova o plano; Ver TELA 04; • P6. O Sistema seta a situação “5” e armazena as informações. |
| Fluxos Alternativos | <ul style="list-style-type: none"> • A01. P5. O Servidor confere e não aprova o plano: <ul style="list-style-type: none"> ◦ A01. P5. a. O sistema mostra opção de inserir observações por categoria; ◦ A01. P5. b. O Servidor insere observações e salva; Ver TELA 05; ◦ A01. P5. c. O sistema seta a situação “3” e salva informações e retorna ao passo P2 do fluxo principal; • A02. P3. O Servidor seleciona o plano “4”; <ul style="list-style-type: none"> ◦ A01. P5. a. O sistema busca as informações e apresenta; ◦ A01. P5. b. O Servidor aprova; Ver TELA 05; ◦ A01. P5. c. O sistema seta a situação “5” e salva informações e retorna ao passo P2 do fluxo principal; |
| Exceções | |

APÊNDICE I – 1387 Atividade 2 e 7 -
Criação dos Casos de Teste e Testes de
Aceitação Executados

| GURI Modulo Plano de Trabalho | | | | | | Resultado esperado | Resultado Obtido | Observações | |
|-------------------------------|--------------------------------|----------------|--------------------------|--|---|---|---|------------------|---|
| Código | Projeto | ID Caso de Uso | Caso de Uso | Objetivo do teste | Pré-condições | Cenário | Resultado esperado | Resultado Obtido | Observações |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Inserção de Dados com sucesso | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para adicionar um novo plano. | O sistema deve exibir as informações do servidor (Nome, idade, cargo, local de exercício, Carga Horária) e salvar as informações adicionadas. | OK | |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Inserção de Dados - Deixar dados obrigatórios em branco | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para adicionar um novo plano. | O sistema deve informar que os campos devem ser preenchidos e não salvar as informações. | | Esta é uma validação desajável, porém não é obrigatória |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Inserção de Dados - retornar os dados corretos do servidor | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para adicionar um novo plano. | O sistema deve buscar e exibir os dados que correspondem ao servidor logado na hora de adicionar um plano. Os dados a serem exibidos são os seguintes: (Nome, idade, cargo, local de exercício, Carga Horária). | OK | |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Inserção de Dados | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para adicionar um novo plano. | | | |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Editar Plano com sucesso | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para editar o plano cadastrado. | O sistema deve salvar as informações alteradas, onde todas as alterações devem ser adicionadas e nunca substituídas. | OK | |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Editar Plano - Deixar dados obrigatórios em branco | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para editar o plano cadastrado. | O sistema deve informar que os campos devem ser preenchidos e não salvar as informações. | | Esta é uma validação desajável, porém não é obrigatória |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Editar Plano - - retornar os dados corretos do servidor | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para editar o plano cadastrado. | O sistema deve buscar e exibir os dados que correspondem ao servidor logado na hora de adicionar um plano. Os dados a serem exibidos são os seguintes: (Nome, idade, cargo, local de exercício, Carga Horária). | OK | |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Clonar Plano com sucesso | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para clonar um plano de trabalho. | O sistema deve duplicar a situação do plano de trabalho selecionado e permitir que o usuário faça a edição do mesmo. | NÃO | Só pode clonar planos do ano anterior. |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Enviar para a chefia com sucesso | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para enviar o plano de trabalho a chefia. | O sistema solicita a confirmação do usuário e troca a situação do plano para "Em avaliação". | OK | |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Enviar para a chefia - trocar valor da situação | O usuário "Servidor" está logado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para enviar o plano de trabalho a chefia. | O sistema deve setar a situação como "Em avaliação". | OK | |
| 1328 | GURI - PTR - Plano de Trabalho | CDU-002 | Manter Plano de Trabalho | Confirmar submissão do plano | O usuário "Servidor" está logado e ter um plano adicionado. | O usuário acessa o menu Administrativo -> Recursos Humanos -> Plano de Trabalho -> Planos de Trabalho. O usuário clica no botão para enviar o plano de trabalho a chefia. | O sistema deve solicitar a confirmação da submissão do plano e encaminhar para a chefia. Alterar o status. | OK | |

Referências

- BARTIE, A. *Garantia da qualidade de software*. [S.l.]: CAMPUS - RJ, 2002. Citado na página 20.
- DESHMUKH, O.; KAUSHIK, M. A overview of software verification and validation and selection process. *International Journal of Computer Trends and Technology*, v. 4, n. 2, 2013. Citado na página 26.
- FAGAN, M. E. Design and code inspection to reduce errors in program development. *IBM Systems Journal*, v. 15, p. 182–211, 1986. Citado na página 21.
- FERREIRA, B.; MOITA, G. F. Inspection technique for the validation of software development processes. *Materials Science and Engineering 10*, 2010. Citado na página 26.
- BOURQUE, P.; FAIRLEY, R. E. (Ed.). *SWEBOK: Guide to the Software Engineering Body of Knowledge*. [S.l.], 2004. Disponível em: <<http://www.computer.org/portal/web/swebok>>. Citado 2 vezes nas páginas 15 e 19.
- IEEE. *IEEE Standard for Software Reviews and Audits*. 2008. Citado 2 vezes nas páginas 21 e 22.
- IFINEDO, P.; NAHAR, N. Quality, impact and success of erp systems: A study involving some firms in the nordic-baltic region. *Journal of Information Technology Impact*, v. 6, n. 1, p. 19–46, 2006. Citado na página 25.
- JONES, C. *Applied Software Measurement: Global Analysis of Productivity and Quality*. [S.l.]: McGraw-Hill Education, 2008. (McGraw Hill professional). ISBN 9780071643863. Citado na página 22.
- KOBROSLY, W.; VASSILIADIS, S. A survey of software functional testing techniques. *Southern Tier Technical Conference*, p. 127–134, 1988. Citado na página 26.
- MYERS, G. J. *The Art of Software Testing*. 2. ed. [S.l.]: John Wiley Sons, 2004. Citado na página 15.
- PEZZÈ, M.; YOUNG, M. *Teste e Análise de Software: Processos, Princípios e Técnicas*. Bookman. ISBN 9788577803743. Disponível em: <<http://books.google.com.br/books?id=aldOmo1oF2AC>>. Citado na página 21.
- PRESSMAN, R. *Engenharia de software*. [S.l.]: McGraw-Hill, 2006. Citado na página 19.
- PRESSMAN, R. S. *Engenharia de Software: Uma abordagem profissional*. [S.l.]: McGraw Hill Brasil, 2011. Citado 2 vezes nas páginas 18 e 20.
- SINDE, A.; HO, C. Independent verification and validation. *Railway Engineering - Challenges for Railway Transportation in Information*, 2008. Citado na página 26.

SOMMERVILLE, I. *Engenharia de Software*. [S.l.]: Pearson Addison, 2007. Citado 5 vezes nas páginas 15, 19, 21, 22 e 29.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. [S.l.]: Pearson Prentice Hall - São Paulo, 2011. Citado 2 vezes nas páginas 20 e 22.

WAZLAWICK, R. S. *Engenharia de Software: Conceitos e Práticas*. [S.l.]: Elsevier, 2013. Citado 2 vezes nas páginas 18 e 19.