EDSON VIEIRA ALVES JÚNIOR

GARANTIA DA QUALIDADE DE SOFTWARE EM PROJETOS COM CONTROLADORES LÓGICOS PROGRAMÁVEIS: UMA REVISÃO DE MÉTODOS

Ouro Preto, MG

2023

EDSON VIEIRA ALVES JÚNIOR

GARANTIA DA QUALIDADE DE SOFTWARE EM PROJETOS COM CONTROLADORES LÓGICOS PROGRAMÁVEIS: UMA REVISÃO DE MÉTODOS

Trabalho de conclusão de curso apresentado ao Instituto Tecnológico Vale, como parte dos requisitos para obtenção do título de especialista em Instrumentação, Controle e Automação de Processos de Mineração.

Área de concentração: Automação Industrial

Orientador: Prof. José Manuel Gonzalez Tubio Perez

Ouro Preto, MG

2023

Título: Garantia da qualidade de software em projetos com controladores lógicos programáveis: uma revisão de métodos
Classificação: ( ) Confidencial ( ) Restrita ( ) Uso Interno (x) Pública

Informações Confidenciais - Informações estratégicas para o Instituto e sua Mantenedora. Seu manuseio é restrito a usuários previamente autorizados pelo Gestor da Informação.

Informações Restritas - Informação cujo conhecimento, manuseio e controle de acesso devem estar limitados a um grupo restrito de empregados que necessitam utilizá-la para exercer suas atividades profissionais.

Informações de Uso Interno - São informações destinadas à utilização interna por empregados e prestadores de serviço.

Informações Públicas - Informações que podem ser distribuídas ao público externo, o que, usualmente, é feito através dos canais corporativos apropriados.

Dados Internacionais de Catalogação na Publicação (CIP)

A478g

Alves Junior, Edson Vieira

Garantia da qualidade de software em projetos com controladores lógicos programáveis: uma revisão de método. Edson Vieira Alves Junior... [et al.] - Ouro Preto, MG: ITV, 2023.

66 p.: il.

Monografia (Especialização latu sensu) - Instituto Tecnológico Vale, 2023.

Orientador: José Manuel Gonzalez Tubio Perez

1. Qualidade de Software. 2. Confiabilidade de Software. 3. Testes de Lógica. I. Perez, José Manuel Gonzalez Tubio. II. Título.

CDD.23. ed. 629.892

Edson Vieira Alves Júnior

GARANTIA DA QUALIDADE DE SOFTWARE EM PROJETOS COM CONTROLADORES LÓGICOS PROGRAMÁVEIS: REVISÃO DE MÉTODOS E PROPOSTA DE PROTOCOLO

Trabalho de conclusão de curso apresentado ao Instituto Tecnológico Vale, como parte dos requisitos para obtenção do título de especialista *lato sensu* em [Automação para Processos de Mineração].

Orientador: Prof. D.Sc. José Manuel Gonzalez Tubio Perez

Trabalho de conclusão de curso defendido e aprovado em 01 de junho de 2023 pela banca examinadora constituída pelos professores:

_____

Prof. D.Sc. José Manuel Gonzalez Tubio Perez
Orientador – Instituto Tecnológico Vale (ITV)

_____

Prof. D.Sc. Gustavo Pessin
Membro interno – Instituto Tecnológico Vale (ITV)

_____

Prof. MSc. Thomás Vargas Barsante e Pinto
Membro interno – Instituto Tecnológico Vale (ITV)

Os Signatários declaram e concordam que a assinatura será efetuada em formato eletrônico. Os Signatários reconhecem a veracidade, autenticidade, integridade, validade e eficácia deste Documento e seus termos, nos termos do art. 219 do Código Civil, em formato eletrônico e/ou assinado pelas Partes por meio de certificados eletrônicos, ainda que sejam certificados eletrônicos não emitidos pela ICP-Brasil, nos termos do art. 10, § 2º, da Medida Provisória nº 2.200-2, de 24 de agosto de 2001 ("MP nº 2.200-2").

PROTOCOLO DE ASSINATURA(S)

O documento acima foi proposto para assinatura digital na plataforma Portal de Assinaturas Vale. Para verificar as assinaturas clique no link: https://vale.portaldeassinaturas.com.br/Verificar/9490-D266-441E-F740 ou vá até o site https://vale.portaldeassinaturas.com.br:443 e utilize o código abaixo para verificar se este documento é válido. The above document was proposed for digital signature on the platform Portal de Assinaturas Vale . To check the signatures click on the link: https://vale.portaldeassinaturas.com.br/Verificar/9490-D266-441E-F740 or go to the Websitehttps://vale.portaldeassinaturas.com.br:443 and use the code below to verify that this document is valid.

Código para verificação: 9490-D266-441E-F740

Hash do Documento

25FB22A560D8E50A2C138DF97A3D8D7EE29C5834742D0F99D5FE520DCE97455F

O(s) nome(s) indicado(s) para assinatura, bem como seu(s) status em 05/06/2023 é(são) :

☑ José Manuel Gonzalez Tubio Perez (Signatário) - em 05/06/2023 12:50 UTC-03:00
Tipo: Assinatura Eletrônica
Identificaçao: Por email: jose.perez@itv.org

Evidências

Client Timestamp Mon Jun 05 2023 12:50:06 GMT-0300 (Horário Padrão de Brasília)
Geolocation Latitude: -21.7513364 Longitude: -41.3310205 Accuracy: 8161.357189291826
IP 187.114.118.162
Hash Evidências:
CCC823468213929BE197C3EB6F1C775B8D05BE55B5A88B084E4330DBCDBF6A2F

☑ GUSTAVO PESSIN (Signatário) - 939.084.900-49 em 05/06/2023 10:32 UTC-03:00
Tipo: Assinatura Eletrônica
Identificaçao: Por email: gustavo.pessin@itv.org

Evidências

Client Timestamp Mon Jun 05 2023 10:33:07 GMT-0300 (Horário Padrão de Brasília)
Geolocation Latitude: -20.3976099 Longitude: -43.5086338 Accuracy: 11.484
IP 200.195.46.98
Hash Evidências:
A18D0F2CA1762FA03A3DB9D3C32E1362F561BDB5D8C824ACFA803B3337101801

☑ Thomás Vargas Barsante e Pinto (Signatário) - 115.302.536-16 em 05/06/2023 10:30 UTC-

03:00
Tipo: Assinatura Eletrônica
Identificaçao: Por email: thomas.pinto@itv.org

Evidências

Client Timestamp Mon Jun 05 2023 10:30:45 GMT-0300 (Horário Padrão de Brasília)
Geolocation Latitude: -19.8681332 Longitude: -43.9830169 Accuracy: 11.479
IP 191.185.78.68
Hash Evidências:
1DB008498ADF2EC728ADA2751ED32EEA1F27653A15BBED19ABEDE3353F5DCAD2

Aos meus pais, irmão e Maíra, amada companheira.

“A leitura do mundo precede a leitura da palavra.”

(Paulo Freire)

RESUMO

Este trabalho foi desenvolvido com o objetivo de demonstrar a aplicação do conceito de Garantia da Qualidade de Software às lógicas de programação de Controladores Lógicos Programáveis (CLPs) por meio do emprego coerente de métodos e instrumentos mais estritamente associados ao tema Confiabilidade de Software. Realizou-se uma pesquisa de abordagem qualitativa, caracterizada como descritiva. O universo de pesquisa levou em conta aplicações de CLPs comumente aplicados à processos de Beneficiamento Mineral e suas respectivas lógicas de programação. A amostra desta pesquisa é composta por um estudo de caso abordando um caso de comissionamento de um Transportador de Correia em determinado processo de beneficiamento mineral. O escopo do referencial teórico abordou a aplicação do conceito de Qualidade de Software aplicado a CLPs nas mais diferentes plataformas e linguagens de programação. Por meio do aprofundamento nos métodos e instrumentos associados à Confiabilidade, pôde-se averiguar a importância de uma adequada estruturação de um roteiro de testes ajustado às características preponderantes das lógicas em desenvolvimento e seus objetivos.

Palavras-chave: Qualidade de Software, Confiabilidade de Software, Testes de lógica.

Fase da Cadeia: Manutenção.

ABSTRACT

This work was developed with the objective of demonstrating the application of the concept of Software Quality Assurance to the programming logic of Programmable Logic Controllers (PLCs) through the consistent use of methods and instruments more strictly associated with the Software Reliability theme. A research with a qualitative approach was carried out, characterized as descriptive. The research universe took into account applications of PLCs commonly applied to Mineral Processing processes and their respective programming logic. The sample of this research consists of a case study addressing a case of commissioning of a Belt Conveyor in a certain mineral processing process. The scope of the theoretical framework addressed the application of the concept of Software Quality applied to PLCs in the most different platforms and programming languages. By deepening the methods and instruments associated with Reliability, it was possible to ascertain the importance of properly structuring a test script adjusted to the preponderant characteristics of the logic under development and its objectives.

Keywords: Software Quality, Software Reliability, Logic Tests.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ITV: Instituto Tecnológico Vale

CLP: Controlador Lógico Programável

t/h: Toneladas por hora (unidade de medida)

SUMÁRIO

1 INTRODUÇÃO

Os sistemas automatizados na mineração já são realidade há décadas, ocupando lugar de destaque em relação aos mais diversos fatores elementares de segurança e confiabilidade destes processos industriais a que se dedicam ao longo de toda a cadeia produtiva.

Estes sistemas, em sua grande maioria compostos por um ou mais controladores interligados em rede, possuem como elemento comum suas lógicas de programação - Algoritmos dedicados à execução necessária de rotinas de inicialização e configuração do sistema e, principalmente, ao adequado atendimento das regras e condições definidas para controle do processo em questão.

Seja em casos de implantações de grandes projetos a partir do zero, de expansões de processos existentes, alterações de grande porte ou mesmo em situações de mudanças corriqueiras de lógicas já em produção em atendimento às equipes de operação e manutenções, pelo forte caráter de interdependência e multidisciplinaridade associados, é fundamental que durante o desenvolvimento destes algoritmos respeitem estritamente aos respectivos papéis em termos de responsabilidades técnicas e de gestão de atividades, visando garantir a correta definição de requisitos funcionais e sua correta implementação.

Este trabalho foi desenvolvido com o objetivo de demonstrar a aplicação do conceito de Garantia da Qualidade de Software às lógicas de programação de CLPs por meio do emprego coerente de métodos e instrumentos relacionados ao tema Confiabilidade de Software.

Assim, esta pesquisa foi dividida em cinco capítulos, em que: o primeiro capítulo foi dedicado a função de apresentação do problema da pesquisa, seu objetivo geral e objetivos específicos, bem como a justificativa para sua elaboração. No segundo capítulo será apresentado o referencial teórico abordando os principais conceitos sobre Garantia de Qualidade de Software, Confiabilidade de Software, Estratégia de Testes, dentre outros. No terceiro capítulo será apresentada a metodologia utilizada para a realização da pesquisa. No quarto, será realizada a conclusão e as considerações finais, de modo a estabelecer relação entre os resultados declarados e os objetivos definidos.

1.1 Objetivo Geral

Demonstrar a aplicação do conceito de Garantia da Qualidade de Software às lógicas de programação de CLPs por meio do emprego coerente de métodos e instrumentos relacionados ao tema Confiabilidade de Software.

1.2 Objetivos Específicos

- Identificar os pontos mais relevantes associados ao conceito de garantia da qualidade de *software* a serem aplicados às lógicas de programação de CLPs.
- Apresentar de forma estruturada métodos, técnicas e conceitos relevantes e aplicáveis a serem adotados como instrumentos na busca por práticas de desenvolvimento de *softwares* mais seguros e confiáveis.
- Demonstrar a relevância do tema em se considerando a criticidade das funções executadas pelos CLPs nos processos de beneficiamento mineral.

1.3 Justificativa

Este trabalho tem como foco o emprego da garantia da qualidade de *software* em lógicas de programação de CLPs que possuem como função principal o amplo controle de equipamentos e processos que compõe os processos de Beneficiamento Mineral.

Segundo classificação da Norma Regulamentadora 4 do Ministério do Trabalho e Emprego (2008), atividades de mineração possuem risco grau 4, o mais elevado nesta escala, os quais, segundo trabalhos de diversos autores, destacam-se:

- Ruído, podendo causar perdas irreversíveis de audição;
- Incêndios e explosões associados ao acondicionamento e manuseio de combustíveis, lubrificantes e explosivos;
- Acidentes gerais severos associados à operação de máquinas e equipamentos de grande porte, havendo possibilidade de causar cortes, prensamentos, esmagamentos, amputações, queimaduras e a morte.

Em se considerando elevado nível de automatização presente nos processos de beneficiamento mineral responsável pelo controle de máquinas e equipamentos desde a frente de lavra até o carregamento da matéria produzida, este trabalho visa estabelecer uma visão

aprofundada acerca da Confiabilidade de *Software* em relação às lógicas de programação do CLPs dedicados a tais processos.

Enquanto contribuição acadêmica e profissional, este trabalho abordará o tema sobre um espectro pouco explorado em se considerando ambientes industriais de alto risco.

Em âmbito organizacional, tal abordagem demonstrará métodos e elementos relevantes aplicáveis a diferentes cenários em que se é exigido o desenvolvimento ou modificação de lógicas de programação, em termos de implantação de projetos e alteração de lógicas já em produção.

2 REFERENCIAL TEÓRICO E FUNDAMENTAÇÃO CIENTÍFICA

Neste capítulo serão apresentados conceitos estruturantes referentes à Gestão da Qualidade de *Software*, assim como os principais elementos que a compõe e mais fortemente se associam com sua aplicabilidade às lógicas de programação de CLPs.

2.1 Qualidade de *Software*

De maneira sintética Pressman (2002) apresenta que Qualidade de *Software* pode ser entendida como uma definição que enfatiza a satisfação de requisitos funcionais e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas que são esperadas em todo *software* desenvolvido profissionalmente. Contudo, de maneira complementar, declara que a Qualidade de *Software* é uma mistura complexa de fatores que vão variar com cada aplicação diferente e com os clientes que as encomendam, donde se enfatizam três pontos:

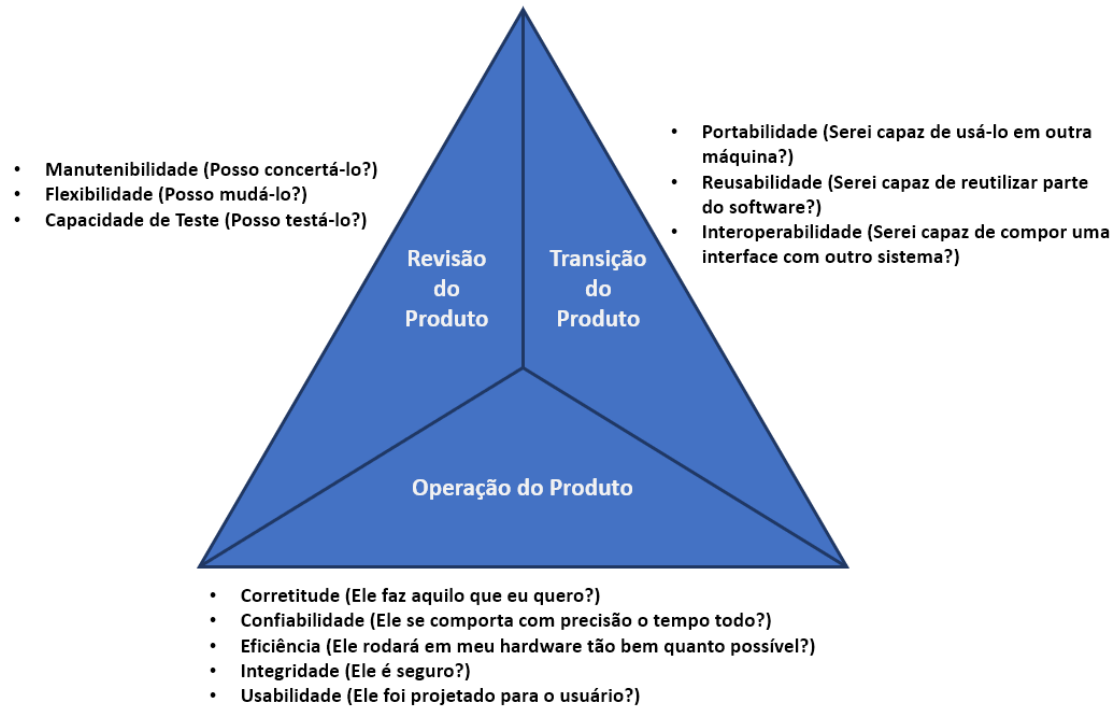
- Requisitos de *Software* são a fundação a partir da qual a qualidade é medida. Falta de conformidade com os requisitos é falta de qualidade.
- Normas especificadas definem um conjunto de critérios de desenvolvimento que guiam o modo pelo qual o *software* é construído. Se não são seguidos critérios, quase sempre vai resultar em falta de qualidade.
- Há um conjunto de requisitos implícitos que frequentemente não são mencionados (p. ex., o desejo de facilidade de uso). Se o *software* satisfaz seus requisitos explícitos, mas deixa de satisfazer os requisitos implícitos, a qualidade do *software* é suspeita.

2.2 Fatores da Qualidade do *Software*

De acordo com Pressman (2007), os fatores que afetam a qualidade do *software* podem ser categorizados em dois grupos amplos: fatores que podem ser medidos diretamente (como no caso de erros e unidades de tempo) e fatores que podem ser medidos apenas indiretamente (como a Usabilidade e a Manutenibilidade).

McCall et al. (1977) propuseram uma categorização dos fatores que afetam a qualidade de *software*. A estes fatores, apresentado na Figura 1, destacam-se três faces importantes: suas características operacionais, sua manutenibilidade de mudanças e sua adaptabilidade a novos ambientes.

Figura 1: Fatores da qualidade de *software* de McCall.



Fonte: PRESSMAN, 2007.

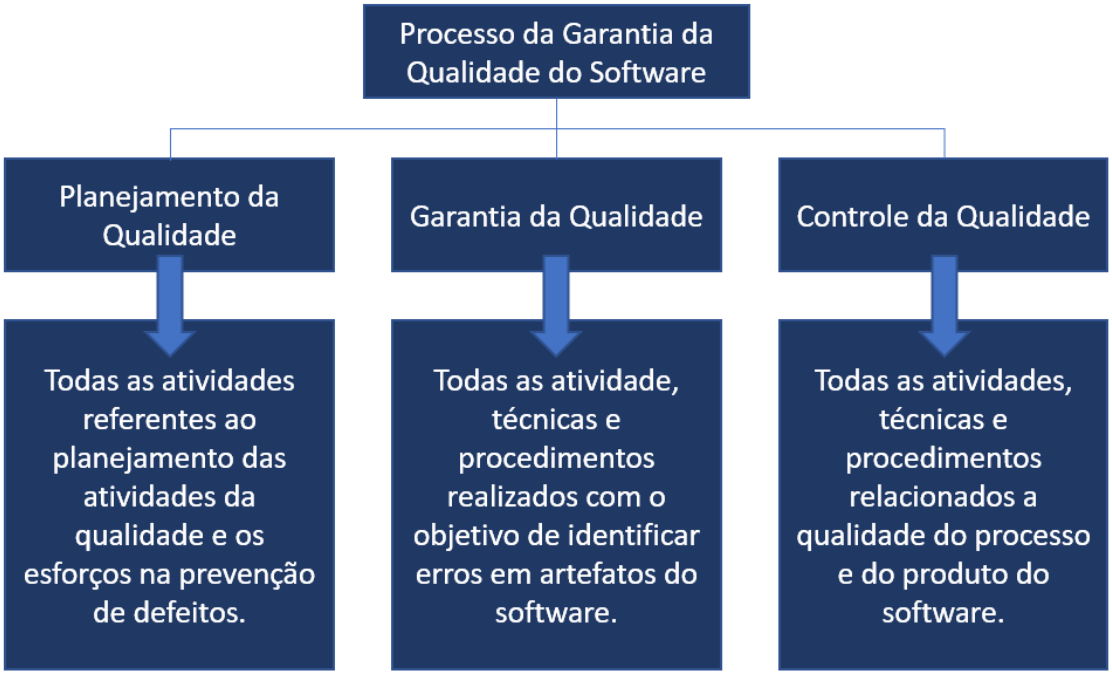
O conceito acima estabelecido por McCall et al. (1977) pressupõe que é difícil, e em certos casos impossível, desenvolver medidas diretas dos fatores de qualidade acima. Contudo guardadas as devidas ponderações de cada um dos fatores, uma expressão composta por cada

um dos fatores úteis à avaliação da qualidade do *software* em questão poderia vir a ser útil na determinação de uma referência de controle de processos de desenvolvimento.

2.3 Os Pilares da Qualidade de *Software*

Dando avanço ao processo de estruturação do conceito da Qualidade de *Software*, Bartié (2002) sugere referência aos trabalhos realizados no PMI (*Project Management Institute*) que organizou o chamado PMBOK (*Project Management Body of Knowledge*) no qual o processo de gerenciamento da qualidade é subdivido em três subprocessos complementares, conforme apresentado na Figura 2 a seguir:

Figura 2: Os pilares da Qualidade de *Software*.



Fonte: BARTIÉ, 2007.

Neste processo, o Planejamento da Qualidade tem como finalidade identificar quais os padrões de qualidade são importantes para o desenvolvimento de projetos e determinar como alcançá-los. Já o processo de Garantia da Qualidade visa garantir o adequado desempenho das etapas que o compõe, satisfazendo aos padrões elegidos. Por fim, o Controle da Qualidade foca

o monitoramento do desempenho dos resultados do projeto para determinar o grau de atendimento aos padrões, critérios e definições existentes.

2.4 Fatores ofensores ao controle de qualidade

Dentre os inúmeros fatores que podem vir a degradar os processos de qualidade de *softwares* indicados por Bartié (2002), destacam-se no ambiente industrial os seguintes aspectos:

- Ausência de gerência de qualidade independente;
- Ausência de procedimentos de testes automatizados;
- Qualidade aplicada tardiamente no desenvolvimento;
- Ausência de profissionais capacitados em Qualidade de *Software*;
- Falta de um modelo corporativo de qualidade;
- Foco em testes progressivos, desconsiderando potenciais efeitos nocivos causados pelas implementações às funcionalidades anteriormente existentes;
- Deficiência no planejamento dos testes;
- Sob pressão, os testes são sacrificados;
- Ausência de um ambiente de testes isolado: O processo de validação de um *software* exige um ambiente exclusivo de testes, isolado de qualquer interferência dos demais ambientes de desenvolvimento e produção;
- Transferir o planejamento e/ou execução dos testes ao desenvolvedor do sistema: O analista de sistemas tem uma tendência natural de produzir testes válidos, ou seja, aqueles já suportados pela aplicação. Dessa forma, ele não tentará submeter o *software* a condições que ele tem consciência que o aplicativo não terá êxito.

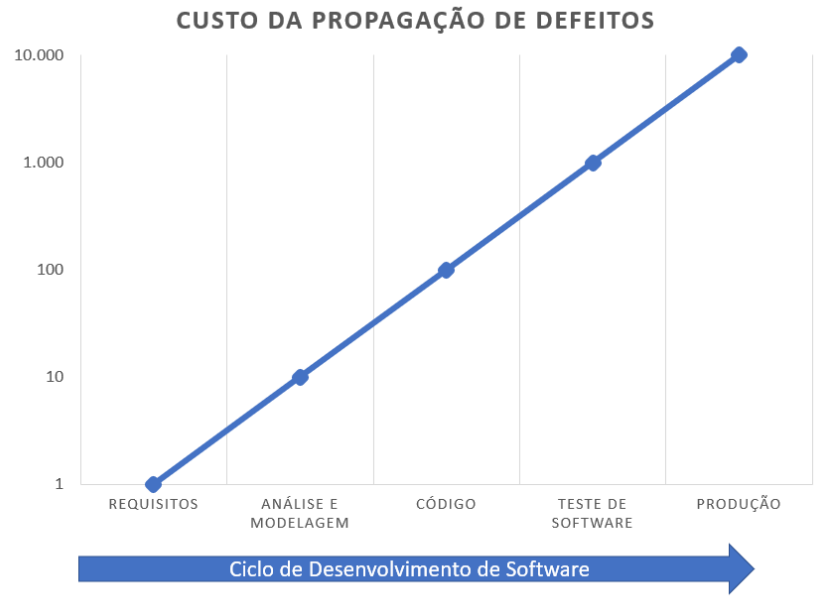
Também segundo Bartié (2002), um dos erros mais comuns sobre qualidade é o modo como este é inserido dentro do processo de engenharia de *software*. Todos tendem a pensar o desenvolvimento de *software* como uma linha do tempo. Dessa forma, definimos metodologias e processos de trabalho para produzir *software* e garantirmos que as etapas serão cumpridas e o produto terá seu ciclo completo de desenvolvimento. O erro é acreditar que dentro desse processo existe um período alocado especialmente para a realização dos testes.

2.5 O custo da propagação de erros

Bartié (2002) afirma que qualquer tipo de erro gera custo financeiro à organização. Tais erros, uma vez identificados durante a etapa de implantação, geram custos de retrabalho, resultando em esforço adicional de remodelagem, recodificação e testes. Contudo, uma vez não identificado e presente no software em produção certamente gerará em determinado momento prejuízos ligados a interrupções na produção ou até mesmo poderá vir a agregar riscos significativos à pessoas e equipamentos dentro e fora dos limites da companhia.

Segundo estudos de Myers, “quanto mais tardiamente descobrimos os erros, mais caros estes se tornam”. Tal conceito fundamenta a chamada “regra de 10” aos custos da correção de erros, onde se pressupõe que, quando um erro não é identificado, os custos de sua correção se multiplicam por 10 a cada fase em que o erro migra. Conforme demonstrado na Figura 3, o custo de tratativa de determinado erro na fase inicial de desenvolvimento de determinado programa teria o suposto valor de uma unidade. À medida que tal erro não seja identificado e tratado, passa a custar 10 vezes mais já na próxima fase de análise e modelagem e assim sucessivamente.

Figura 2: Custo da propagação de defeitos.



Fonte: O autor.

É importante destacar que a exposição acima toma uma base numérica para representação de custo. Contudo, em processos caracterizados como de alto de risco como no caso dos processos de mineração, os impactos de determinadas falhas podem vir a gerar sérios transtornos e perdas de vidas humanas.

2.6 Requisitos de Software

Uma compreensão completa dos requisitos de *software* é fundamental para um bem-sucedido desenvolvimento de *software*. Não importa quão bem projetado ou quão bem codificado seja, um programa mal analisado e especificado desapontará o usuário e trará aborrecimentos ao desenvolvedor (PRESSMAN, 2007).

Segundo Sommerville (2007), os requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e suas restrições operacionais. Esses requisitos refletem a necessidade dos clientes de um sistema que ajudam a resolver algum problema, por exemplo, controlar um dispositivo, enviar um pedido ou encontrar informações.

Pressman (2007) complementa indicando que a análise e especificação de requisitos pode parecer uma tarefa relativamente simples, mas as aparências enganam. O conteúdo de comunicação é muito elevado. Abundam as chances de interpretações errôneas e informações falsas e a ambiguidade é provável.

Pressman (2007) declara que os requisitos de *software* são a base a partir da qual a qualidade é medida e, portanto, a falta de conformidade aos requisitos significa falta de qualidade. No mesmo sentido, também define que a especificação, independentemente do modo pelo qual a realizamos, pode ser vista como um processo de representação e, em última análise, as exigências devem ser representadas de uma forma que leva à implementação bem-sucedida.

2.7 Princípios da Especificação

Transformar em código especificações ruins, incompletas e imprecisas acarreta produtos não adequados e pouco confiáveis. Dentre os oito princípios propostos por Zelkowitz (1989), 5 deles são destacados abaixo, em se considerando a realização de uma boa especificação de requisitos:

1. Separe funcionalidade de implementação: A especificação é uma descrição daquilo que é desejado e não de como deve ser realizado (implementado);
2. Uma linguagem de especificação de sistemas orientada ao processo é exigida: Ou seja, a especificação deve ser abrangente o suficiente para abarcar a todos os componentes que compõe o sistema em questão;
3. Uma especificação deve abranger o ambiente no qual o sistema opera;
4. Uma especificação de sistema deve ser um modelo cognitivo, devendo descrever um sistema da forma como ele é percebido por sua comunidade de usuários;
5. Uma especificação deve ser operacional.

2.8 Testes que Garantem a Qualidade do Produto

De acordo Bartié (2002), a qualidade dos produtos de *software* pode ser garantida através de sistemáticas aplicações de testes nos vários estágios do desenvolvimento da aplicação. Esses testes são conhecidos como testes de validação porque, quando construímos uma unidade de *software*, validamos sua estrutura interna e sua aderência aos requisitos estabelecidos. Avaliamos sua integração com as demais unidades de *software* existentes, validando as interfaces de comunicação existentes entre os componentes de *software*. Quando determinado subsistema ou mesmo a solução está finalizada, validamos a solução tecnológica como um todo, submetendo a testes de todas as categorias possíveis.

Quando se objetiva validar determinada solução, subconscientemente, cria-se um ambiente de máxima concordância com as condições elaboradas. Contudo, se o objetivo de quem avalia a lógica de programação for provar o contrário, ou seja, sua não conformidade, passa-se a ser capaz de identificar os inúmeros cenários negativos possíveis.

Bartié (2002) então conclui que devemos ter em mente que os testes podem ser usados para mostrar a presença de erros, mas nunca sua ausência. A razão óbvia para essa conclusão é o fato de que um processo de qualidade, por melhor que ele seja, não conseguirá cobrir todas as infinitas combinações existentes em um ambiente de execução real.

2.8.1 Elementos e estruturação de Testes

Segundo Bartié (2002), o processo de desenvolvimento de *software* pressupõe dois momentos bem definidos. O primeiro é a fase de coleta de informações de negócios e o

planejamento da arquitetura do *software*. Essa fase caracteriza-se pelo esforço em criar um perfeito entendimento entre o negócio a ser atendido e o *software* a ser construído, como se estivéssemos criando uma grande maquete do projeto de *software* – é nesse primeiro momento que uma forma básica de teste se destaca: os testes de Verificação.

A outra etapa posterior é dependente do processamento de dados por meio de infraestrutura computacional. Para esta etapa se aplicam testes para verificação da lógica de programação desenvolvida quanto à sua correspondência com os parâmetros e definições declaradas. Este método é definido como teste de Validação.

2.8.2 Testes de Verificação

Bartié (2002) define que os testes de verificação podem ser entendidos como um processo de auditoria de atividades e avaliação de documentos gerados durantes todas as fases do processo de engenharia de *software*. A principal característica dos testes de verificação é o fato de não envolver o processamento de *softwares*. Na verdade, estas atividades antecedem a criação do aplicativo, exatamente para garantir que todas as decisões e definições foram adequadamente entendidas e aceitas pelos diversos grupos que integrarão o processo de desenvolvimento.

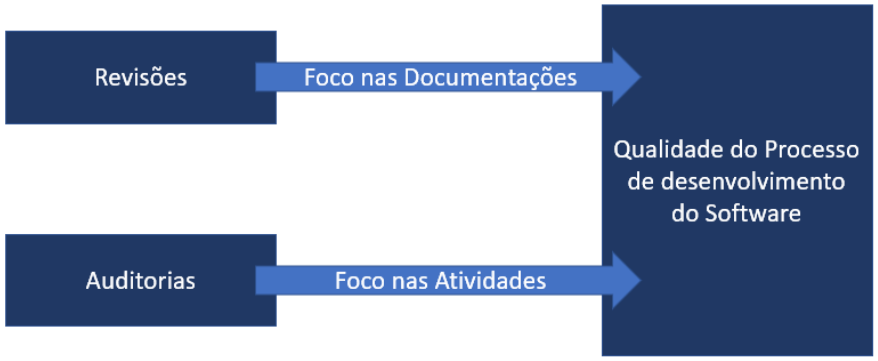
Deste modo, se faz possível determinar se as atividades em evolução estão sendo executadas a contento, bem como a geração da documentação necessária para formalização de seu desenvolvimento, considerando:

- Uniformização das atividades: Estruturar os trabalhos do revisor, treiná-los e adotar instrumentos e mecanismos de controle como listas de verificação, os famosos checklists, representam ações fundamentais para garantir a padronização das atividades.
- Comunicação e coesão da equipe: Comunicação acerca de atualizações, frequência dos encontros, mantendo informados níveis hierárquicos superiores aos dos diretamente envolvidos no desenvolvimento, de modo a estimular a participação, contribuição e coesão do grupo.
- Abrangência, análise de efetividade e retroalimentação das revisões: Em situações em que o processo de desenvolvimento das lógicas de programação ocorra em meio a outras atividades diversas, como, por exemplo, a montagem e instalação de equipamentos em

campo, é importante garantir que os indivíduos dedicados a verificação das documentações acompanhem continuamente a evolução das ações realizadas, de modo a oportunizar a retroalimentação entre as etapas e diferentes disciplinas envolvidas.

Os testes de verificação devem abranger e ter ação durante todas as etapas do desenvolvimento da aplicação. Para tanto, o processo de verificação não deve se limitar apenas as fases iniciais de desenvolvimento por meio das revisões, mas sim, se manter envolvida com o mesmo nível de comprometimento durante a execução das atividades de desenvolvimento por meio de auditorias periódicas.

Figura 4: Processos de verificação.



Fonte: BARTIÉ, 2007.

Segundo Bartié (2002), há alguns pontos críticos na avaliação da qualidade dos documentos, a saber:

- Completo: Todos os requisitos documentados;
- Simples e Objetivo: Essência e propósito facilmente compreensíveis;
- Preciso: Não deve haver margem para duplas interpretações;
- Consistente: Não haver conflito entre requisitos definidos;
- Relevante: Possuir relação com escopo determinado;
- Testável: Nível de informações disponíveis deverão ser suficientes de modo a permitir se quantificar se os requisitos foram implementados em conformidade;
- Factível: De implementação viável em termos econômicos, humanos e tecnológicos.

2.8.3 Testes de Validação

De acordo com Bartié (2002), os testes de validação podem ser entendidos como um processo formal de avaliação de produtos tecnológicos que podem ser aplicados em componentes isolados, módulos existentes ou mesmo nos sistemas como um todo. Seu objetivo é avaliar a conformidade do *software* com os requisitos e especificações analisadas e revisadas nas etapas do projeto. O autor indica os seguintes aspectos principais associados a este tipo de teste:

- "Contrário aos testes de verificação, que avaliam as documentações e atividades com o objetivo de detectar erros ou quebras do processo, os testes da validação possuem agora um produto computacional que pode ser executado";
- "Cabe aos profissionais voltados à realização dos testes de *software* buscarem todos os meios e recursos possíveis para criar infraestrutura que possibilite identificar o maior número de erros, gerando o menor esforço possível";
- "Os testes de validação têm por objetivo identificar o maior número possível de erros, tanto nos componentes isolados, quanto na solução tecnológica como um todo";
- "O sucesso da validação está apoiado diretamente em um forte planejamento de todas as atividades de testes, nas quais a concentração dos trabalhos de validação nos componentes mais complexos e nos requisitos mais críticos contribui para aumentar a eficiência dos procedimentos de detecção de defeitos, uma vez que esses componentes concentram a maior quantidade de erros".

O avanço dos testes é dado por meio da conclusão das fases de desenvolvimento que se inicia com os testes em fragmentos de código ou funções isoladas para posteriormente vir a ser verificadas após serem integrados, até chegar ao ponto de serem estruturados em módulos. Ao final do cumprimento destas etapas, a solução então é submetida ao aceite pelos clientes e usuários.

Tomando como cenário um processo de beneficiamento mineral, podemos citar como exemplos de testes de validação a atuação e monitoramento das respostas de cada um dos canais de um novo módulo de entradas digitais instalado em uma remota de campo; a atuação de uma sirene de partida em um transportador de correias; o comportamento de um sensor de nível

instalado em um tanque de polpa, estando o tanque vazio e, posteriormente, o tanque totalmente cheio; dentre outros.

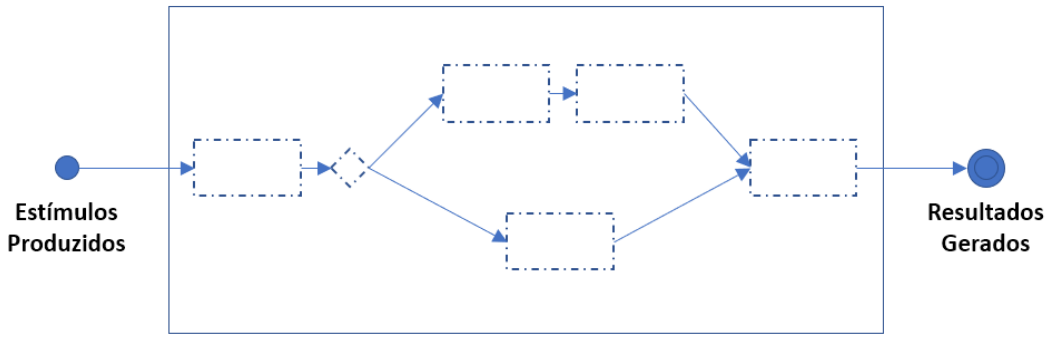
2.8.4 Estratégias Fundamentais de Testes

"Os testes de validação envolvem a execução total ou parcial do *software* a ser avaliado, porém existem duas únicas abordagens para conduzir um processo de validação: pela Estratégia Caixa Branca, ou pela Estratégia Caixa Preta". Por razões de ajuste terminológico, nesta obra os testes Caixa Preta e Caixa Branca serão tomados por Caixa Opaca e Caixa Transparente, respectivamente.

2.8.4.1 Estratégia Caixa Opaca

Por meio desta estratégia é possível realizar a validação de requisitos do sistema com base em suas definições funcionais declaradas, não se atendo a como o algoritmo foi estruturado e o processamento dos estímulos realizados são executados.

Figura 5: Visão de testes caixa preta [caixa opaca]



Fonte: BARTIÉ, 2007.

Dentre as vantagens desta estratégia, devemos considerar a oportunidade de o responsável pela execução dos testes de estar em contato profundo com os requisitos funcionais do sistema, de modo a ser capaz de definir e reproduzir os estímulos relevantes ao processo em questão, bem como identificar e apurar os resultados gerados durante o teste. Posto isso, recomenda-se fortemente que Testes Caixa Opaca sejam realizado em conjunto com os responsáveis pelas definições de processo e/ou projeto, de modo que o nível de conhecimento

dos requisitos aplicado seja o mais elevado possível tendo em vista a validação do atendimento aos requisitos e critérios de processo.

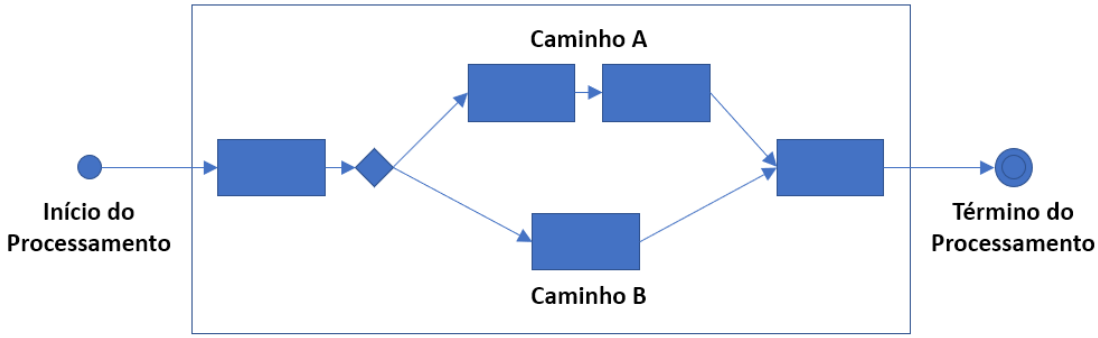
Segundo Bartié (2002), “o grande desafio em se implantar o método [de teste] nas organizações é convencer a área que já executa as atividades de homologação e começar a exigir um planejamento mais apurado e transparente para que todas as outras áreas possam ter acesso a todos os cenários de testes que estão sendo executados”.

2.8.4.2 Estratégia Caixa Transparente

Testes Caixa Transparente se baseiam na arquitetura interna do *software*. Como define Bartié (2002), esses testes empregam técnicas que objetivam identificar defeitos nas estruturas internas dos programas através da simulação que exercitem adequadamente todas as estruturas utilizadas na codificação.

De implementação mais complexa, para adequada execução de Testes Caixa Transparente, se faz necessário que o profissional responsável por sua execução possua considerável domínio da arquitetura interna da lógica de programação. Além do fator humano, Testes Caixa Transparente comumente demandam maior tempo para serem executados em sua plenitude, o que, em muitas das situações, pode vir a ser ameaça a essa estratégia de teste.

Figura 6: Visão de testes de caixa branca [caixa transparente]



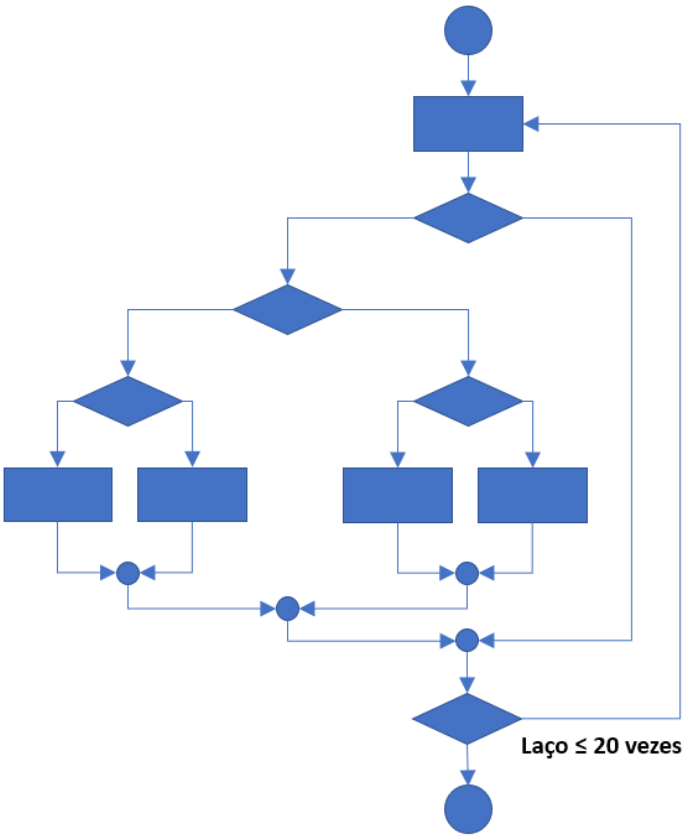
Fonte: BARTIÉ, 2007.

Pressman (2007) ressalva que, “numa primeira observação, poderia parecer que um teste de Caixa Branca [Caixa Transparente] efetuado muito cuidadosamente levaria a ‘100% de programas corretos’”. Infelizmente, testes exaustivos apresentam certos problemas logísticos,

porque, mesmo para pequenos programas, o número de caminhos lógicos possíveis pode ser muito grande. O projeto procedimental ilustrado pela Figura 7 poderia corresponder a um programa de 100 linhas com um único laço que pode ser executado não mais do que 20 vezes. Neste caso, portanto, conclui-se que haveria aproximadamente impressionantes 10^{14} caminhos possíveis a serem executados.

Neste cenário é importante se ressaltar a elevadíssima complexidade de previsão de todas as situações possíveis associadas às entradas, bem como os modos de falha possíveis ou, minimamente, mais frequentes, o que, sem dúvida representará uma ameaça à estratégia de testes e à confiabilidade da solução como um todos.

Figura 7: Problemas com testes exaustivos.



Fonte: Pressman, 2007, p. 792.

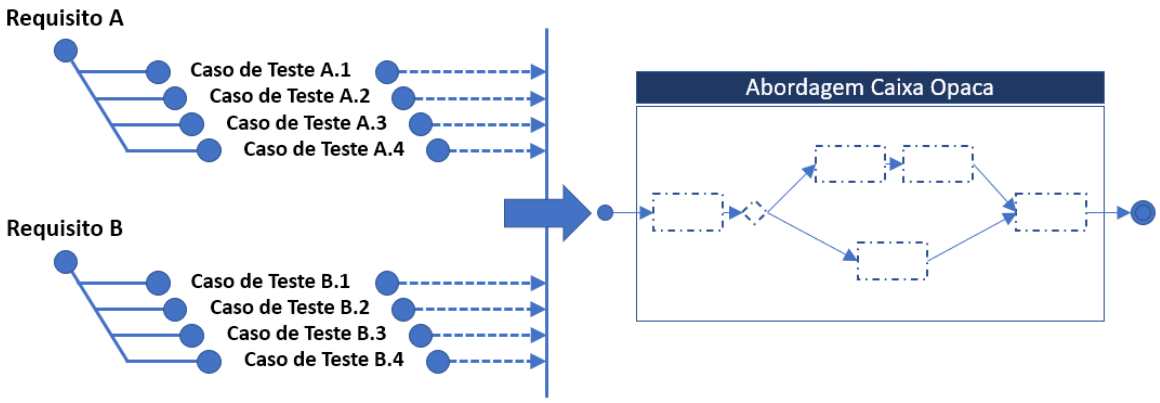
Pressman (2007) complementa que a estratégia Caixa Transparente não deve ser desprezada como pouco prático, pois um número limitado de caminhos lógicos importantes pode ser selecionado e exercitado.

2.8.5 Casos de Testes associado aos Métodos das Caixas

De acordo com Sommerville (2007), “o projeto de casos de testes é parte do teste de sistemas e de componentes, no qual você projeta os casos (entradas e saídas esperadas) que testam o sistema. A meta do processo do projeto de casos de teste é criar um conjunto de casos de testes eficazes para descobrir defeitos do programa e demonstrar que o sistema atende aos requisitos”. Sommerville (2007) ainda complementa que “para projetar um caso de teste, selecione uma característica do sistema ou do componente que você vai testar. Depois selecione um conjunto de entradas e saídas para executar aquela característica, documente as saídas esperadas ou faixas de saídas e, quando possível, projete uma verificação automática que teste se as saídas reais e as saídas esperadas coincidem”.

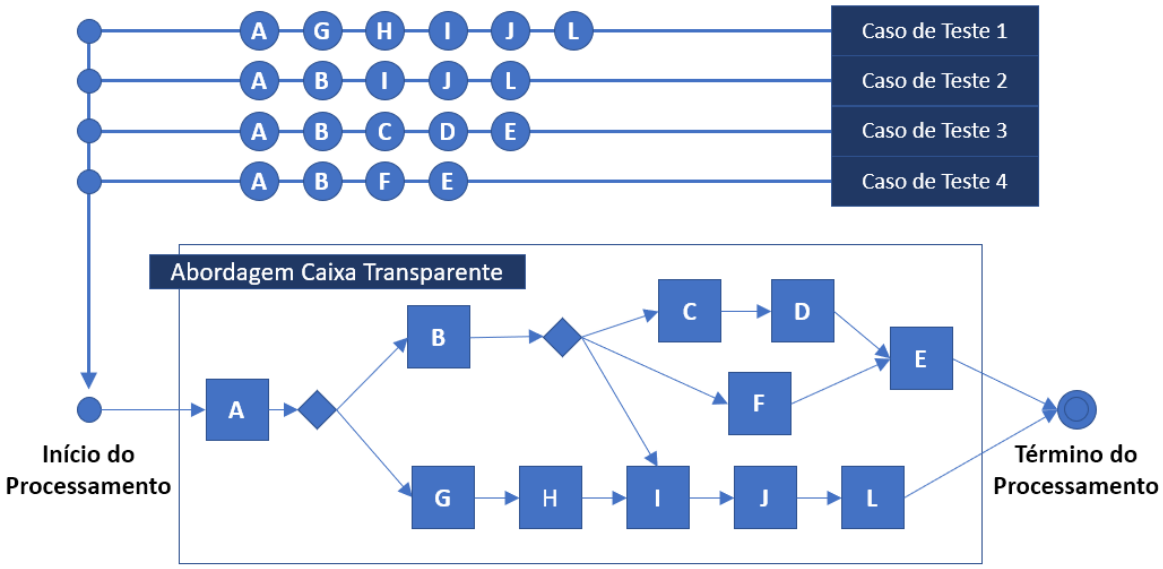
A partir deste ponto, os conceitos das Caixas Opacas e Transparentes e de Casos de Testes se fundem, de modo a gerar variações representadas nas figuras 8 e 9 a seguir.

Figura 8: Técnicas caixa preta [caixa opaca] para obtenção de casos de testes.



Fonte: BARTIÉ, 2007.

Figura 9: Técnicas caixa branca [caixa transparente] para obtenção dos casos de testes



Fonte: BARTIÉ, 2007.

Dados os conceitos de Caixa Opaca e Transparente, devemos sempre ter em mente que tais estratégias não são concorrentes entre si, mas sim, concomitantes e complementares. Cabe ao responsável pela definição do Roteiro de Testes definir em maior ou menor grau os itens onde cada estratégia deverá ser aplicada em maior ou menor grau a depender das características da demanda e o nível de maturidade e confiança associado às funções e estruturas internas do programa. Conforme declara Pressman (2007), os atributos de ambas as estratégias "podem ser combinados para oferecer uma abordagem que valide a interface com o *software* e garanta seletivamente que o funcionamento interno do *software* esteja correto".

Além da definição dos requisitos e elementos a serem verificados e suas respectivas estratégias associadas, caberá ao responsável pela condução dos testes definir os casos de testes mais eficazes na identificação dos defeitos e relevantes na determinação do adequado atendimento aos requisitos funcionais do programa.

A partir deste ponto, já se faz possível elaborar minimamente o Roteiro de Testes a ser aplicado na Validação da aplicação. É recomendado que tal documento seja submetido ao

conhecimento dos demais envolvidos responsáveis pela implantação em caráter multidisciplinar, de modo a garantir sua maior eficácia possível.

A predominância dos Testes Caixa Opaca em relação à estratégia de Testes Caixa Transparente, em boa parte dos casos, será determinada pelo nível de maturidade e confiança nas funções necessárias à implementação. Deste modo, por exemplo, no caso de alterações do sistema relativas a expansões do processo limitadas à inclusão de equipamentos de mesmo tipo aos dos já operativos, é esperado e adequado que haja um maior número de requisitos de validação baseados em Teste Caixa Opaca em detrimento aos relativos à Caixa Transparente.

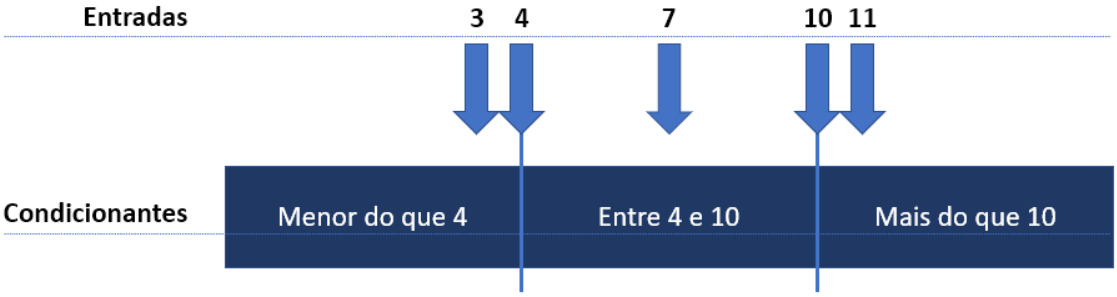
2.8.6 Cenários de Testes

Conforme afirma Bartié (2002), "um dos maiores desafios de um processo de garantia da qualidade é conseguir medir o grau de qualidade alcançado nos testes de *software*". Se em nosso entendimento o maior volume de casos de testes significa o maior número de cenários adequadamente simulados e garantidos, então devemos buscar todas as alternativas possíveis e inseri-las em nosso processo de teste de *software*, de forma a refinar e ampliar o nível de cobertura alcançado.

Das possíveis abordagens adequadas à definição dos Casos de Testes, destacam-se:

- Orientada aos Requisitos: Casos de Teste são estabelecidos de modo que sejam capazes de demonstrar que o sistema cumpre a cada requisito funcional estabelecido. Neste caso, o principal referencial a ser considerado deverá ser o Memorial Descritivo do projeto em questão, ou documento similar que se preste a declarar de forma mais detalhada possível condições e funções mínimas necessárias para o adequado funcionamento e operação do processo.
- Estruturada em Partições: Casos de Teste são definidos com base na identificação de valores e regiões de valores específicos associados aos valores de entrada e saída do sistema. Cada uma destas partições possui características e objetivos específicas, como no caso da interação com números negativos, limiares operacionais definidos por condições operacionais de comparação, valores que extrapolam suas respectivas faixas e limites operacionais etc.

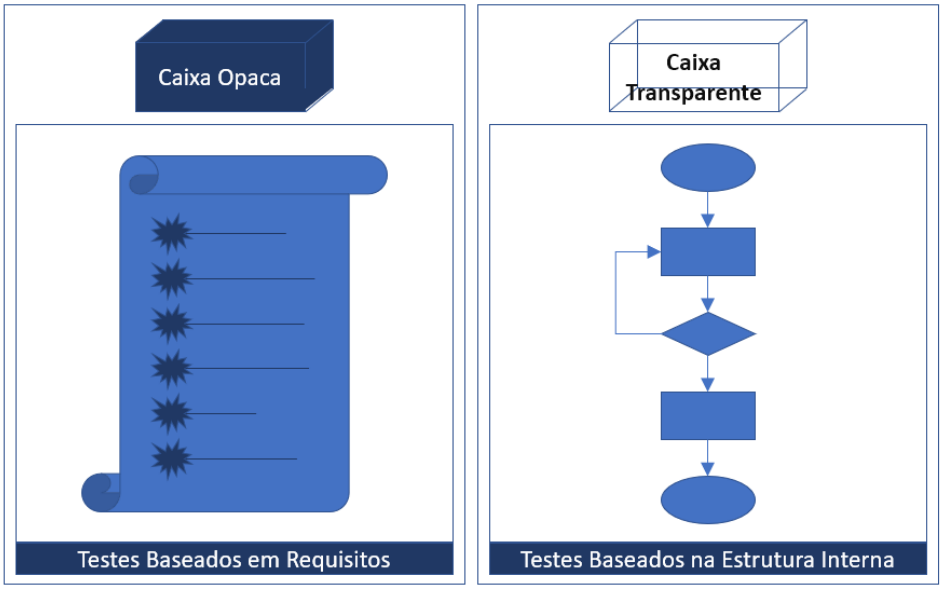
Figura 10: Definição de cenário de teste estruturado em partições.



Fonte: O autor.

- Teste Estrutural: Casos de Teste são definidos com base na estrutura adotada no desenvolvimento do programa, tendo como objetivo garantir a máxima abrangência possível na busca por defeitos.

Figura 11: Definição de cenário de teste estrutural.



Fonte: O autor.

2.8.7 Método de Decomposição de Requisitos

De acordo com Bartié (2002), os Requisitos direcionam todo o processo de desenvolvimento de *software*. Cada requisito de *software* carrega consigo um conjunto de cenários possíveis dentro de uma realidade do sistema que irá atendê-lo. Didaticamente, podemos decompor cada requisito em três cenários distintos:

- Cenário Primário: Cenário ótimo e mais provável de execução da lógica, sem exceções às regras definidas.
- Cenários Alternativos: Demais cenários previstos, compostos por possíveis variações em relação ao Cenário Primário, como no caso em que há falha nos sensores e/ou atuadores de campo, erros operacionais prováveis ou mudanças no comportamento do processo, dentro dos limites de capacidade dos equipamentos.
- Cenário de Exceção: Possíveis cenários agregados de problemas e inconsistências que impossibilitam o processamento pleno de determinada condição associada a um determinado requisito que podem vir a ser entendidos como uma falha crítica no CLPs, indisponibilidade nas redes de comunicação ou do sistema supervisório da planta, dentre outros.

A decomposição de um requisito em cenários é fundamental para explorarmos todas as possibilidades envolvidas na dinâmica do *software*, exercitando nossa visão para além das condições mais prováveis de utilização e/ou relativas ao ambiente de execução.

2.8.8 Identificação de Categorias de Testes

Bartié (2002) nos diz que “muitas empresas e seus profissionais têm dificuldade em distinguir as diversas motivações que levam à realização dos testes de *software*. É claro que todos os testes têm por objetivo a identificação de erros, porém devemos entender que a localização de todo e qualquer tipo de defeito exigirá um esforço muito grande da equipe de testes. Se não temos tempo nem recursos suficientes para executar todos os procedimentos de testes, devemos planejar uma estratégia na qual estabelecemos quais tipo de erros queremos prioritariamente encontrar”.

Em linha com a constatação declarada acima e reiterando o caráter consideravelmente limitado de recursos disponíveis na etapa de testes, saber identificar e definir adequadamente

as categorias de Testes de *Software* com que trabalharemos durante a execução dos Testes certamente contribuirá muito com a viabilidade da estratégia de atuação e eficiência das análises. Bartié (2002) destaca que se o "levantamento focar uma única categoria de testes, poderemos melhor conduzir o processo, possibilitando concentrar nossas energias em uma única perspectiva de defeitos a serem identificados". Dentre as diversas categorias possíveis podemos citar:

- Segurança: Em se tratando de Automação aplicada à processos industriais, o quesito segurança tem por objetivo detectar defeitos na lógica de programação que podem vir a gerar dano a pessoas ou danos materiais significativos.
- Funcional: Tem por objetivo abranger a todas as condições relevantes ao atendimento dos requisitos funcionais, conforme definidos nos documentos de especificação.
- Usabilidade: Objetiva avaliar as condições de utilização da lógica de programação levando em consideração a perspectiva de seus usuários, como no caso dos Técnicos Operacionais nas Salas de Controle das unidades produtivas.
- Configuração (ou Parametrização): Tem como objetivo executar o programa considerando as diversas possibilidades de ajustes a que o processo possa vir a se submetido, observando o seu comportamento em relação aos requisitos funcionais declarados.
- Recuperação: Objetivo verificar o comportamento da lógica diante da ocorrência de erro ou ocorrência de condição anormal à operação do processo, como no caso da perda de comunicação entre controladores, ativos de medição e controle, dentre outros.
- Contingência: Tem por objetivo a validação de procedimentos de contingência em caso de falhas críticas do sistema, de modo a avaliar os procedimentos definidos a situações extremas de falhas no sistema, garantindo a continuidade da capacidade produtiva do processo.

A abordagem de Categorização de Testes, além de contribuir na definição de priorização dos elementos a serem testados, possui também a importante capacidade de ampliar a visão da equipe de Testes acerca dos aspectos e impactos potenciais da implementação em desenvolvimento a partir de outros pontos que, em grande parte das ocasiões não são sequer consideradas durante o processo de desenvolvimento e implantação.

2.8.9 Refinamento por Probabilidade de Erro

Segundo Bartié (2002), esse método é um velho conhecido dos desenvolvedores porque está baseado na intuição e experiência de testar condições que normalmente provocam erros. Produz bons resultados, pois concentram alta probabilidade de identificação de defeitos no *software*. Um histórico de origens de erros bem montado poderá destacar quais situações são as mais recorrentes e ampliar o conjunto de itens mais problemáticos.

A seguir, destaca-se alguns dos erros mais comuns durante o desenvolvimento de programas:

- Tabelas vazias ou nulas;
- Indisponibilidade de dados de entrada;
- Erros causados por condições estabelecidas apenas em sua primeira execução;
- Valores de entrada inválidos ou negativos.

2.8.10 Critérios de Finalização

“Um dos aspectos mais importantes a serem estabelecidos durante os procedimentos de Qualidade é estabelecer os critérios de finalização dessas atividades, ou seja, determinar quais serão os indicadores que determinarão se determinado documento, atividade ou *software* alcançou o nível de qualidade desejado”.

“Os critérios de finalização servirão como uma referência aos grupos de qualidade para aplicarem os procedimentos e somente darem por encerrada uma atividade quando esses critérios forem plenamente satisfeitos. É muito importante estabelecer que em todas as atividades de qualidade existirá ao menos um critério de finalização definido”.

3 PROTOCOLO DE CONFIABILIDADE DE *SOFTWARE*

3.1 Objetivo

O Protocolo de Confiabilidade de *Software* visa estabelecer um modelo metodológico viável de testes de lógicas de programação de CLPs dedicados à processos industriais. Por meio deste protocolo se buscará atingir um nível satisfatório da Garantia da Qualidade de *Software* aplicada à Controladores Programáveis em processos industriais. O desenvolvimento deste modelo proposto tomará como pano de fundo o conceito da Confiabilidade de *Software*, trazendo consigo a todo momento sua questão fundamental: “Ele [o *software*] se comporta com precisão o tempo todo”?

3.2 Campos de aplicação

De modo a ser o mais abrangente possível sem deixar de considerar as características mais relevantes dos cenários onde os testes deverão ser aplicados, o trabalho aqui exposto propõe dois procedimentos principais, a saber:

a) Procedimento aplicado a um caso de implantação de projetos;
b) Procedimento aplicado a um caso de mudança de lógica existente, já operativa.

Desta forma é esperado que as situações mais frequentes e relevantes em termos de testes sejam consideradas, bem como sejam expostos métodos e instrumentos úteis à abordagem da Confiabilidade de *Software*.

3.2.1 Caso de Implantação de Projetos

3.2.1.1 Características gerais

No caso do desenvolvimento e implantação de projetos é preciso considerar a complexidade adicional associada aos testes pela necessidade de programação de elementos lógicos elementares ao funcionamento de equipamentos e processos.

Portanto, tendo em vista a Confiabilidade de *Software* requerida para a garantia dos níveis de segurança de pessoas e máquinas, caberá ao responsável pela implantação identificar o nível de desenvolvimento requerido para a implantação do projeto, estabelecendo um número

maior de fases e processos de verificação e validação conforme o grau de desenvolvimento exigido.

Dentre as etapas de programação em situações em que há a necessidade de desenvolvimento de lógicas em níveis mais elementares (ou, em outras palavras, “do zero”), destaca-se o desenvolvimento de Blocos de Funções, também conhecidos como Blocos Típicos ou Blocos Padrão. Tais blocos, além de desempenharem funções críticas em termos de partida e parada, atuação de defeitos e intertravamentos de equipamentos, também se caracterizam pela elevada quantidade de implementações necessárias para atendimento aos requisitos de projeto pelo fato de estarem, geralmente, associados à ativos bastante comuns aos processos industriais como válvulas e motores. Em grande parte dos casos, é comum se utilizar blocos já desenvolvidos e operacionalizados em outras unidades, contudo, em ocasiões em que não exista nenhuma referência já operacional disponível para atendimento à demanda, se faz necessário desenvolver um novo bloco específico para a aplicação. Pela junção de tais aspectos, visando garantir a confiabilidade do sistema e se afastar do risco de retrabalho, deve-se dedicar máximo critério na definição de conceitos e minucia na realização das baterias de testes de verificação.

3.2.1.2 Papéis e responsabilidades

Durante o desenvolvimento e implantação de projetos é comum haver interação entre equipes com diferentes propósitos e atribuições dentro de todo o processo de desenvolvimento, implantação e operacionalização. No caso da implantação de projetos, será considerado um cenário mediano em relação ao desenvolvimento em se considerando que são raros os casos em que viremos a nos deparar com a necessidade de iniciarmos totalmente do zero, sem referência alguma em termos de estrutura de projeto, associado a equipamentos e processos preexistentes.

Dado este cenário, conforme apresentado na tabela 1, temos dada a relação entre equipes, papéis e responsabilidades esperada:

Tabela 1: Papéis e responsabilidades em implantação de projetos

EQUIPE	PAPEL	RESPONSABILIDADE
Engenharia Básica	Definir técnicas e serviços a serem contratados para atendimento às definições de conceito.	Identificar com precisão as características do que será construído, levando em consideração impactos sociais, humanos e ambientais do empreendimento.
Fabricante / Fornecedor	Construir e fornecer máquinas e equipamentos necessários à composição do projeto.	Fabricar equipamentos conforme os parâmetros e características gerais declaradas pelo projeto.
Implantação	Executar a montagem, instalação e operacionalização do processo determinado.	Executar obras e atividades necessárias à implantação do projeto de modo a atender às definições das etapas antecessoras e premissas apresentadas pelas equipes responsável pela utilização e manutenção do negócio.
Manutenção	Manter equipamentos, processos e sistemas em adequado funcionamento após a entrega do projeto	Manter a disponibilidade dos equipamentos de modo a eliminar seus defeitos, mantendo os padrões de qualidade dos produtos.
Operação e Processo	Utilizar a planta instalada dentro dos limites de projeto e normas vigentes	Utilizar o processo de maneira adequada, buscando aprimorar continuamente sua eficiência, produtividade e qualidade.

Fonte: O autor.

Apesar da distinção notada entre os papéis e responsabilidades das diferentes equipes que compõe o processo de Implantação de Projetos em questão, destaca-se a importância da comunicação e sinergia entre os grupos nas mais diversas etapas de desenvolvimento das atividades. O envolvimento e estabelecimento de compromissos em relação à normas, padrões e boas práticas das equipes de Operação e Manutenção da planta certamente garantirão condições de sustentabilidade e desenvolvimento muito superiores após estabelecido o início das operações do processo.

Neste sentido, em relação à Garantia da Qualidade de *softwares* de CLPs, cabe relevar a relação entre as equipes de Implantação de Projetos e de Manutenção de Automação.

3.2.1.3 Requisitos mínimos

3.2.1.3.1 Estrutura base do projeto (programa de CLP) a ser tomada como referência

Projeto desenvolvido anteriormente para atendimento à outra demanda de onde se possa aproveitar estruturas e elementos lógicos úteis ao desenvolvido do projeto em questão.

3.2.1.3.2 Descritivo Funcional / Memorial Descritivo

Documento de engenharia elaborado com o objetivo de descrever todas as regras e funções associadas aos equipamentos que compõe o projeto em questão em termos de

intertravamentos, grupos e sequências de partida e parada, automatismos, níveis de alarmes, dentre outros. Tal documento deve levar em consideração parâmetros técnicos, operacionais e de segurança de equipamentos e processos, declarando em detalhes como o sistema de controle deverá reagir e interagir com seus operadores e demais sistemas integrados.

3.2.1.3.3 Diagrama P&ID, Unifilares e Multifilares

Diagramas P&ID (do inglês "*Piping and Instruments Diagram"*, ou seja, Diagrama de Tubulação e Instrumentos) têm como objetivo estabelecer de maneira uniforme e padronizada a identificação de instrumentos e sistemas de instrumentação, sendo muito úteis durante a definição de estratégias de malhas de controle e suas respectivas implementações. De maneira análoga, Diagramas Uni/Multifilares buscam representar esquemas de interligação de dispositivos elétricos.

3.2.1.3.4 Relação de Entradas e Saídas

Como o próprio nome indica, a Relação de Entradas e Saídas é responsável por declarar a função de cada uma das entradas e saídas (analógicas ou discretas) relativas ao sistema de controle adotado. É esperado que cada um dos pontos relacionados seja acompanhado minimamente do módulo ou cartão a que pertence; seu endereço, conforme padrão de programação definido pelo fabricante do CLP; nomenclatura no processo que deverá ser utilizado ao ser programado na lógica, conforme padrão de tageamento aplicável e descrição resumida de sua finalidade no processo.

3.2.1.3.5 Arquitetura de Redes (Redes de Controle e/ou Redes Industriais)

Documento dedicado à representação geral das redes de comunicação entre dispositivos que compõe o sistema de controle, podendo vir a se comunicar por meio dos mais diversos protocolos implementados no ambiente industrial. Por meio destes documentos se faz possível identificar parâmetros relevantes à programação de CLPs como a localização dos dispositivos em seus respectivos segmentos, endereço de rede dos dispositivos, identificação da presença de Repetidores, Conversores de Mídia e *Gateways* (Conversores de Protocolo). A partir de correta identificação dos elementos que compõe as redes, se faz possível a adequada implementação

de importantes rotinas e funções de monitoramento de estado da comunicação, como no caso das funções *watchdog*.

3.2.1.3.6 Manuais técnicos (Mapas de Memória)

Manuais técnicos dos mais diferentes tipos de instrumentos, acionamentos e dispositivos de rede são sempre necessários ao desenvolvimento de projetos e integração de novos dispositivos a sistemas de controle. Por meio destes documentos são disponibilizados seus respectivos Mapas de Memória, responsáveis pela determinação das áreas de endereçamento de variáveis internas do dispositivo. A correta utilização destas informações é de fundamental importância para o uso confiável e seguro dos mais variados dispositivos em comunicação com o sistema de controle.

3.2.1.4 Estratégias de testes recomendadas

3.2.1.4.1 Testes de Verificação

Conforme estabelecido pelo referencial teórico identificado, os testes de Verificação se darão por meio de dois principais meios: Revisões – onde o foco se estabelecerá nas documentações geradas a serem utilizadas – e Auditorias, cujo foco serão as atividades realizadas ao longo do desenvolvimento de todo o projeto.

É natural e esperado que durante o desenvolvimento e implantação de projetos haja uma elevada geração de documentos com as mais diversas finalidades. Uma vez conhecendo quais os documentos úteis à disciplina de Automação, é fundamental que se garanta que as devidas revisões nos documentos a serem tomados como referência tenham sido realizadas por seus elaboradores responsáveis. Neste ponto é importante destacar que há diferentes figuras responsáveis pela geração dos documentos e, além das devidas revisões que se mostrarem necessárias, só devem ser consideradas referências válidas, documentos devidamente Aprovados e Autorizados.

Da mesma forma, devido ao elevado nível de interação entre diferentes disciplinas, caberá à realização sistemática de Auditorias em todas as etapas de desenvolvimento, buscando identificar erros e desvios nos processos obrigatórios para a programação com confiabilidade.

Também muito conhecidas como Checklists, as Listas de Verificação são poderosas ferramentas a serem adotadas nas revisões de documentos e auditorias das atividades. Por meio deste instrumento se faz possível direcionar o foco dos responsáveis pela análise de qualidade dos documentos e identificar falhas na execução de atividades.

3.2.1.4.1.1 Estruturando e aplicando Listas de Verificação para revisão de documentos e auditorias de atividades

Para a elaboração das listas de verificação dedicadas à revisão de documentos é sugerido que sejam considerados os seguintes parâmetros:

- ✓ Verificar se todas as revisões foram devidamente aprovadas e autorizadas, com ênfase na revisão final;
- ✓ Avaliar se todos os equipamentos que compõe o projeto foram contemplados nas definições de programação;
- ✓ Avaliar se os conceitos básicos relativos à disciplina de Automação estão aplicados de forma adequada (Intertravamentos; Defeitos; Alarmes; Malhas de Controle; dentre outros);
- ✓ Verificar se todos os Diagramas P&ID, Unifilares e Multifilares estão disponíveis;
- ✓ Avaliar se todos os protocolos de redes possuem suas respectivas arquiteturas disponíveis;
- ✓ Verificar se Arquiteturas de Redes comtemplam a todos os dispositivos que compõe as redes;
- ✓ Identificar se todos os Manuais Técnicos estão disponíveis e possuem as informações necessárias para desenvolvimento das lógicas de programação;
- ✓ Verificar disponibilidade de padrões e boas práticas a serem adotadas como referência durante o desenvolvimento da programação.

Vale destacar que as listas de verificação dedicadas à auditoria de atividades deverão ser específicas à cada fase do desenvolvimento do projeto, uma vez que as auditorias deverão ser aplicadas do seu início ao fim. Posto isso, seguem abaixo, divididas por etapa,

- Preparação:

  - Há máquinas de desenvolvimento de *software* (notebooks e desktops) em quantidade suficiente para as frentes de serviço definidas?
  - As máquinas de desenvolvimento estão adequadamente configuradas para acesso aos respectivos domínios de redes em que deverão ser empregadas?
  - Todos os usuários que necessitarão de acesso às máquinas de desenvolvimento de *software* possuem as devidas permissões de acesso?
  - Licenças de *softwares* necessários à programação e configuração de dispositivos estão disponíveis?
  - Os s*oftwares* necessários ao desenvolvimento das atividades de programação estão instalados / disponíveis para utilização nas máquinas de desenvolvimento?
  - Os CLPs e demais dispositivos necessários para a execução dos testes de plataforma estão disponíveis?
- Desenvolvimento:
  - O programa base para desenvolvimento do projeto está disponível em sua versão adequada?
  - Todos os documentos de padrões e boas práticas de programação estão disponíveis?
  - Todos os testes de bancada / plataforma definidos como necessários foram realizados?

- Comissionamento:
  - Foram criados todos os tags no sistema de monitoramento de variáveis de processo (PIMS)?
  - Foram realizados todos os testes de validação definidos como necessários para a implantação do projeto (detalhado no capítulo 3.2.1.4.2 – Testes de Validação)?
- Operação assistida:
  - Todas as malhas de controle previstas implementadas foram devidamente sintonizadas?

- Foram validadas todas as estratégias de controle associadas às malhas de controle previstas?
- Foram identificadas

- Finalização:
  - Foram revogadas todas as chaves e permissões de acesso a redes e sistemas dos usuários envolvidos exclusivamente nas atividades de implantação do projeto?
  - Foram realizadas todas as revisões e *as-builds* necessários?
  - Todos os programas desenvolvidos foram entregues em suas versões mais atualizadas, totalmente compatíveis com as versões em processamento nos CLPs?
  - Todos os documentos e programas foram postados em seus respectivos repositórios?
  - Todos os ativos foram devidamente cadastrados nos sistemas de gestão e controle da manutenção (ex.: SAP, IBM Maximo etc.)?
  - Todos os componentes ou recurso dedicado à aquisição de itens sobressalentes foram repassados às equipes de manutenção responsáveis em quantidade adequada?
  - Todos os treinamentos necessários à manutenção dos dispositivos e sistemas de controle foram realizados com o público definido como necessário?
  - Todos os treinamentos necessários à operação da planta em implantação foram realizados com o público definido como necessário?
- Entrega:
  - Todos os documentos e listas de validação (Testes de Validação) foram entregues aos responsáveis / interessados e devidamente preenchidas?
  - Os responsáveis pela operação e manutenção da planta em implantação formalizaram a aceitação do projeto (assinatura do TAP – Termo de Aceitação de Projeto)?

3.2.1.4.2 Testes de Validação

Os testes de validação se caracterizam por submeter as lógicas de programação à determinadas condições de uso de modo que seja avaliado se o comportamento está conforme o definido e esperado pelo projeto e demais equipes responsáveis. Diante de um cenário de implantação de projetos, a combinação de diferentes técnicas e métodos de testes de validação será fundamental para garantir maior abrangência e pertinência em relação às situações com maior potencialidade de erro. Neste capítulo abordaremos as principais características dos testes de validação aplicados em implantação de projetos, considerando suas principais variações em termos de ambiente onde este pode vir a ser realizado, abordando também alguns métodos e ferramentas que podem vir a ser úteis a seu desenvolvimento.

3.2.1.4.2.1 Testes de Bancada / Testes de Aceitação em Fábrica (TAF)

Testes de Bancada ou Testes de Aceitação em Fábrica (TAF) podem ser entendidos como experimentações reproduzidas em uma estrutura isolada do ambiente industrial, cujos equipamentos e dispositivos que compõe sua estrutura são suficientemente equivalentes em termos de especificações técnicas e configurações em relação ao meio onde o produto validado virá a ser inserido.

A decisão por se adotar os testes de bancada se justifica pela necessidade de validação de funções lógicas, dispositivos, equipamentos e sistemas ainda sem referência de implantação disponível ou que caracterizem algum tipo de risco atípico ao processo de implantação em caso de falha. Dentre as mais diversas situações em que se cabe considerar a realização de testes de validação em bancadas, podemos citar: Elaboração de blocos de função lógica a serem reproduzidos em larga escala; definição de configuração de hardware e testes de comunicação com instrumento de medição inédito no processo; integração de novo sistema supervisório; dentre outros.

3.2.1.4.2.2 Testes Ponto a Ponto

Testes Ponto a Ponto visam garantir que todos os componentes do projeto funcionem corretamente juntos e se caracterizam pela verificação individualizada de cada uma das entradas ou saídas de sinais discretos ou analógicos que compõe o sistema. Os testes ponto tem como objetivo verificar se todos os componentes do sistema de automação se comunicam corretamente e se os dados são transmitidos sem perda de informação ou erros.

Tal tipo de teste geralmente requer a interação entre duas ou mais pessoas onde, enquanto uma gera os estímulos necessários aos elementos de campo, outra atesta o comportamento do sistema sob verificação.

Testes Ponto a Ponto usualmente sem aplicam a:

- Testes Ponto a Ponto de sensores e atuadores discretos em campo;
- Testes Ponto a Ponto de transmissores analógicos (preferencialmente condicionando o processo às variações ao longo de toda a escala de medição definida ou, minimamente, simulando a variável a partir do campo);
- Testes de troca de sinais com sistema supervisório;
    - Funções gerais do sistema (navegação; chamada de telas e janelas; *displays* de analógicas; caixas de ajuste; sumário de alarmes etc.);
    - Comandos;
    - Estados;
    - Defeitos;
    - Intertravamentos;
    - Alarmes;
    - Sinalizações em geral.

Devido ao elevado número de sinais que compõe um projeto, os testes ponto a ponto usualmente encontram grande dificuldade em serem executados devido ao elevado tempo e recurso que demandam para serem executados em plenitude.

3.2.1.5 Matriz de Causa e Efeito

A Matriz de Causa e Efeito é um documento usualmente aplicado à validação de funções críticas e de segurança que de maneira bastante objetiva organiza informações sobre diferentes causas e seus efeitos associados ao comportamento esperado de um determinado equipamento ou processo industrial. O principal objetivo do documento é fornecer informações claras ao engenheiro responsável pela programação de lógicas de controle, contudo também pode ser um poderoso aliado ao analista de qualidade de *software* durante a execução de testes de validação realizados.

Devido à sua estrutura, este documento possui a capacidade de representar de forma sintética uma ou várias combinações que podem estar associadas à sensibilização de determinada condição lógica definida pelo projeto. As causas, por exemplo, podem considerar uma mudança no estado de uma entrada digital, a atuação de uma botoeira, o envio de certo comando a partir do sistema supervisório, o atingimento de um determinado nível de alarme associado à uma variável analógica, entre outros comandos ou eventos. Já os efeitos podem incluir o acionamento de um motor ou bomba, a abertura de uma válvula de alívio de pressão, ativação de uma sirene ou sinalização de emergência, a atuação de um intertravamento ou defeito no sistema de controle, assim por diante.

As causas são apresentadas na porção esquerda do documento, enquanto os efeitos são descritos na região superior direita. Ambos são caracterizados por um número sequencial ou seu respectivo tag de processo, agregadas de descrições e dados de referência documental. Na região criada pela interseção das linhas e colunas, são pontuadas as relações entre causas e efeitos declaradas.

Figura 12: Matriz de Causa e Efeito.

		CAUSAS			EFEITOS												
					FECHA VÁLVULA	FECHA VÁLVULA	LIGA BOMBA	DESLIGA BOMBA	.	NOTA 1	NOTA 1	NOTA 1	NOTA 1	.	ESD-1	ESD-2	.
P&ID	TAG	FUNÇÃO	SETPOINT		1	2	3	4	5	6	7	8	9	10	11	12	13
DRW001_XPTO	PAHH_54321	PRESSÃO MUITO ALTA ÓLEO HIDRÁULICO	936	1	x												
DRW001_XPTO	PAHH_54322	PRESSÃO MUITO ALTA ÓLEO HIDRÁULICO	936	2	x												
DRW001_XPTO	PAHH_54323	PRESSÃO MUITO ALTA ÁGUA SELAGEM	440	3		x											
DRW001_XPTO	PAHH_54324	PRESSÃO MUITO ALTA ÁGUA SELAGEM	440	4		x											
DRW001_XPTO	LAL_98765	NÍVEL BAIXO ÓLEO HIDRÁULICO		5				x									
DRW001_XPTO	LAH_98765	NÍVEL ALTO ÓLEO HIDRÁULICO		6			x										
DRW001_XPTO	PALL_54327	PRESSÃO MUITO BAIXA ÓLEO HIDRÁULICO	100	7													
DRW001_XPTO	PALL_23456	PRESSÃO MUITO BAIXA ÓLEO HIDRÁULICO	100	8													
DRW001_XPTO	PALL_23457	PRESSÃO MUITO BAIXA ÁGUA SELAGEM	95	9													
DRW001_XPTO	PALL_23458	PRESSÃO MUITO BAIXA ÁGUA SELAGEM	95	10													
-	-	-	-	11													
DRW001_XPTO	SDV_54333	POSIÇÃO VÁLVULA ÓLEO HIDRÁULICO	-	12						x	x						
DRW001_XPTO	SDV_54334	POSIÇÃO VÁLVULA ÁGUA SELAGEM	-	13								x	x				
-	-	-	-	14													
DRW001_XPTO	ADEF_12345	ALARME PARADA POR DEFEITO	-	15											x		

Fonte: O autor.

Como mencionado, por se tratar de um documento usualmente aplicado a funções críticas e de segurança do processo, a matriz de Causa e Efeito buscar ter a estrutura mais enxuta e objetiva possível, contudo, há também a possibilidade de se utilizar operadores lógicos nas

regiões de interseção entre causas e efeitos de modo a representar relações associadas à múltiplas variáveis, conforme apresentado de maneira simplificada na figura 13.

Figura 13: Matriz de Causa e Efeito agregada de operadores lógicos.

	Output 1	Output 2	Output 3	Output 4	Output 5	Output 6	Output 7	Output 8
Input 1								
Input 2			AND				AND	OR
Input 3		OR				OR	AND	
Input 4	OR		OR		OR			OR
Input 5	OR							
Input 6					AND			
Input 7	AND			OR	AND			OR
Input 8			AND					

Fonte: O autor.

Da figura 13, tomada como exemplo, conclui-se que a saída 1 (*Output* 1) deverá ser atuada sempre que as entradas 4 e 7 (*Input* 4 e *Input* 7) ou as entradas 5 e 7 (*Input* 5 e *Input* 7) sejam atuadas simultaneamente.

3.2.1.6 Definição e aplicação de Casos de Testes

A partir deste ponto, já se faz possível elaborar minimamente o Roteiro de Testes a ser aplicado na Validação da aplicação com base na definição de casos de testes. É recomendado que tal documento seja submetido ao conhecimento dos demais envolvidos responsáveis pela implantação em caráter multidisciplinar, de modo a garantir sua maior eficácia possível.

Em se tratando de um ambiente de implantação de projetos, onde é previsto o desenvolvimento e implementação de funções elementares do sistema, é fundamental que Testes Caixa Transparente sejam amplamente empregados, de modo a garantir a adequada validação de todas as funções estruturais da lógica desenvolvida.

3.2.1.7 Um caso real: Revisão do sistema de frenagem do Transportador de Correias

Durante a etapa de operação assistida do novo processo em implantação em determinada planta de beneficiamento mineral, foi identificado que as pastilhas de freio do sistema de

frenagem de um dos transportadores de correias que compunham as novas instalações vinham apresentando nível de desgaste acima do esperado.

Após uma longa apuração de ajustes e parâmetros em campo pelos responsáveis das equipes de manutenção mecânica não indicar causas em potencial capazes de gerar tal comportamento anormal, buscou-se então avaliar se a lógica de controle do sistema de frenagem havia sido implementada de forma adequada. Neste momento, constatou-se que os freios estavam sempre sendo atuados em sua máxima capacidade, modo este recomendado somente para condições extremas de processo.

Ao buscar referências no documento memorial descritivo do projeto para reprogramação da lógica de controle, não foram identificadas informações suficientes que indicassem de que forma os diferentes níveis de intensidade de frenagem deveriam ser aplicados nas ocasiões de parada do sistema.

A partir de então, um trabalho de revisão geral partindo das definições básicas de projeto envolvendo as equipes de desenvolvimento, implantação e automação foi iniciado. Após algumas sessões de reuniões, o grupo finalmente alcançou uma versão final verificada do documento. Com as devidas aprovações, as novas definições atualizadas foram programadas, verificadas e devidamente validadas após a aplicação de um roteiro de testes também consensado por todos envolvidos.

Este caso nos traz a importância de uma robusta e abrangente aplicação dos métodos focados na garantia de qualidade de *software* por meio da verificação e validação de documentos e lógicas de programação geradas durante o desenvolvimento e implantação de projetos.

3.2.2 Caso de alteração de lógicas de programação já em operação

Seguramente é possível se afirmar que alterações em lógicas de programação já em produção, com ênfase nos processos de mineração, fazem parte da rotina das operações. Dentre os agentes causadores de uma dinâmica de alterações tão elevada, podemos citar:

- Baixa maturidade agregada a fase conceitual dos projetos implantados quanto à possibilidade de utilização dos processos;

- Má utilização das plantas e déficit de manutenção de ativos, resultando em elevado índice de quebras de equipamentos demandando, por consequência, frequentes alterações para ajustes de rotas de processo e/ou adequação de parâmetros operacionais;
- Baixo nível de engenharia aplicada a continuidade dos processos / baixo nível de interação e sinergia entre as disciplinas responsáveis;
- Elevado grau de empirismo (“tentativa e erro”) associado a várias pequenas alterações, sem a devida análise prévia e posterior acompanhamento de ganhos;
- Baixo nível de maturidade do controle de qualidade aplicado aos processos, dificultando a identificação de anomalias, desvios e oportunidades e a replicação de boas práticas;
- Processos altamente suscetíveis a mudanças no cenário externo, ocasionando em profundas alterações conceituais das operações.

Posto isso, a partir deste capítulo, serão apresentados os elementos mais relevantes associados a um protocolo de testes a ser aplicado em casos de alterações de lógicas de programação em processos já em operação.

3.2.2.1 Papéis e responsabilidades

Dada tamanha demanda por alterações em lógicas de programação nos processos de beneficiamento mineral, uma adequada definição de papeis e responsabilidades, capaz de informar e garantir a contribuição técnica de todos os personagens envolvidos, é de fundamental importância para a garantia da confiabilidade das lógicas de programação responsável pelo controle de equipamentos e processos.

Na tabela 2 se encontram declarados em caráter propositivo papéis e responsabilidades a serem definidos em um fluxo de alteração de lógicas de processos já em operação, abrangendo desde a definição do descritivo funcional a ser alterado, até a validação da mudança, após intervenção por parte da equipe de automação.

Tabela 2: Papéis e responsabilidades em alterações de lógicas já em operação

INDIVÍDUO	PAPEL	RESPONSABILIDADE
Solicitante	- Efetuar a solicitação formal da alteração; - Identificar e estabelecer as devidas condições para a realização da alteração; - Validar a alteração junto às demais disciplinas afetadas e responsável pela executante da alteração lógica.	- Conhecer como a lógica opera; - Conhecer os impactos em potencial associados à mudança; - Definir detalhadamente as condições a serem alteradas; - Avaliar e definir de maneira conjunta com os demais indivíduos envolvidos e responsáveis pelas demais disciplinas afetadas com a alteração o roteiro de testes a ser aplicado para validação da mudaça; - Diligenciar o planejamento e programação das ações e recursos definidos; - Validar formalmente alteração após sua conclusão.
Aprovadores	- Avaliar tecnicamente os impactos em potencial associados à mudança nos termos das respectivas disciplinas a que são responsáveis;	- Aprovar ou reprovar a solicitação da mudança em solicitação; - Apoiar tecnicamente as definições estabelecidas, roteiro de testes e processo de validação da alteração.
Executantes	- Executar a alteração da lógica	- Executar a alteração da lógica de programação conforme definido, respeitando aos padrões, normas e boas práticas associadas ao projeto afetado; - Suportar a todas as ações definidas para testes e validação da mudança.
Analista de testes	- Executar testes de verificação e validação pertinentes;	- Garantir a confiabilidade da aplicação alterada com base nos padrões, normas e boas práticas aplicáveis; - Garantir e execução plena do roteiro de testes estabelecido pela equipe multidisciplinar

Fonte: O autor.

3.2.2.2 Dificultadores e restrições às verificações e validações

As alterações de lógicas realizadas em processos de beneficiamento mineral já em operação trazem consigo particularidades, em larga medida, que tornam o processo de Garantia da Qualidade de *Software* ainda mais complexa. Dentre estes principais fatores agravantes, podemos citar:

- Tendência de minimização da relevância das lógicas de programação nos processos: Devido a elevada recorrência de alterações em lógicas de programação motivadas pelos mais variados motivos, acaba-se por criar um ambiente de banalização deste tipo de intervenção, caracterizando assim, um ambiente de informalidade desprovido de controles eficazes. Em situações como estas, tranquilamente é possível afirmar que os testes de verificação e validação são os primeiros a serem sacrificados;

- Necessidade de as alterações serem realizadas com o processo em operação em boa parte dos casos: Tal condição de atuação requer máxima atenção e domínio de quem executa a alteração, dificultando significativamente a atuação do analista de testes;
- Ausência de definição formal em relação à forma como o processo está sendo controlado até o momento: Antes de definir onde se quer chegar é fundamental saber onde se está. É por meio desta afirmação trivial que se revela um problema bastante comum nos processos de beneficiamento mineral. Seja dado pela ausência de documentação ou pela pouca experiência das pessoas envolvidas nos processos em que estão inseridos, a clara definição dos descritivos funcionais para mudanças em processos já em operação se torna em muitos casos um desafio significativo às equipes de operação, processo, manutenções e automação;
- Baixo domínio dos padrões e práticas de programação por parte dos executantes das alterações e / ou analistas de testes;
- Elevada pressão psicológica sobre os executantes em função do risco de interrupção nas operações durante as intervenções e elevada carga de serviço;
- Falta ou insuficiência de uma análise de riscos prévia de forma se identificar os impactos em potencial e definir os marcos de validação relevantes a alteração.

Devido à forte tendência por se omitir os testes durante alterações em lógicas de programação já em operação, definir adequadamente um roteiro de testes, bem como conhecer e determinar seus impactos diretos e em potencial desde o início do processo de definição da solicitação de alteração da lógica devem passar a ser considerados um dos principais requisitos obrigatórios para solicitação de mudanças. Desta forma, espera-se alcançar um maior nível de consciência e cooperação entre os indivíduos envolvidos e responsáveis pela confiabilidade das lógicas de programação e dos processos a que se dedicam.

3.2.2.3 Estratégias de testes recomendadas

Assim como abordado para os casos de implantação de projetos, as recomendações de testes para os casos de alterações de lógicas em produção serão apresentadas variando a partir de dois tipos principais: Verificação (coleta de informações de negócios e o planejamento da arquitetura do *software*) e Validação (dependente da existência de elemento computacional).

É importante frisar que, a depender das características das alterações em lógicas já em operação, podemos vir a tomar pose de muitas das estratégias e elementos recomendados ao caso de implantação de projetos. Deste modo, este capítulo buscará não se aprofundar na declaração detalhada dos diversos tipos de testes e instrumentos de verificação e validação, mas sim, se concentrará em destacar aqueles que se mostram mais relevantes e pertinentes de adoção.

3.2.2.3.1 Testes de Verificação

Devido ao menor volume de documentação envolvida nas alterações de lógicas de programação já em operação, é esperado uma menor presença destes tipos de testes. Além da atenção às recomendações trazidas no capítulo 3.2.1.4.1.1. – Estruturando e aplicando Listas de Verificação para revisão de documentos e auditorias de atividades –, nestas ocasiões de solicitações de alterações de lógicas de programação é importante que seja minimante avaliada a sensibilização de procedimentos e instruções operacionais pela mudança, bem como documentos de engenharia existentes, procedendo junto aos respectivos responsáveis por tais documentos as devidas revisões, comunicações e retreinamentos.

3.2.2.3.1.1 Atuação com dupla verificação

Levando em consideração a definição de que aquele que executa alteração da lógica não deve ser o mesmo indivíduo que a avalia e atesta sua confiabilidade, recomenda-se planejar as atividades que envolvam alterações de lógicas considerando a participação de dois indivíduos com competências equivalentes. Deste modo, por meio de atuação com dupla verificação (realizada pelo próprio executante e pelo analista de testes), pode-se garantir, minimamente, a realização da busca por erros nas lógicas em alteração, garantindo um maior nível de confiabilidade nas intervenções.

3.2.2.3.2 Testes de Validação

Testes de validação por natureza consomem muito tempo e grande quantidade de mão de obra e por este aspecto tendem a enfrentar resistência ainda maior por parte daqueles interessados pela implementação de alterações em lógicas em ambientes operativos. Diante de tal cenário, ter uma visão clara das estratégias e instrumentos úteis à garantia da confiabilidade da solução é aspecto fundamental para a argumentação e defesa pela realização dos testes definidos como necessários.

Por vezes, pequenas alterações em determinadas lógicas de uma usina de beneficiamento mineral, podem implicar em inúmeras alterações de comportamento de outras lógicas e processos a esta condição inicial associados. Portanto, tão importante quanto a definição e defesa da realização dos testes adequados é a consciência de que, exceto casos extremamente simples, nenhuma alteração é passível de ser totalmente (100%) testada. De modo a atenuar os riscos em potencial extraídos desta constatação, é fundamental aos responsáveis pela garantia de qualidade das lógicas estarem amparados por um fluxo formal das alterações abrangente, envolvendo pessoas que possuam compromisso com a segurança dos processos e o domínio necessário de suas respectivas disciplinas potencialmente afetadas.

3.2.2.3.2.1 Listas de Verificação

Não diferente das demais situações de testes a serem aplicadas, as listas de verificação (*checklists*) têm um papel extremamente relevante na organização da execução dos testes.

É recomendado que tais listas estejam separadas por tipo de variável ou função sob análise, como por exemplo: Defeitos, intertravamentos, sinalizações, alarmes, variáveis discretas, variáveis analógicas, sequências de partida/desligamento, funções de automatismo, malhas de controle, dentre outros. Através desta distinção é esperado que os devidos parâmetros e informações a serem avaliadas pelo analista de testes possam ser registradas e validadas.

3.2.2.3.2.2 Matriz de Causa e Efeito

Às funções críticas, relacionadas à segurança de pessoas e equipamentos, é recomendada a adoção das matrizes de causa e efeito.

3.2.2.3.2.3 Casos de Testes e o Método das Caixas

Situações em que já se tenha domínio e confiança em relação à estrutura interna da programação, como geralmente ocorre nas alterações de lógicas em produção, é esperado o predomínio de Testes do tipo Caixa Opaca no roteiro estabelecido.

É fortemente recomendado que a definição dos casos de testes para a busca de erros seja realizada de maneira conjunta com os demais responsáveis pelas disciplinas sensibilizadas pela mudança de modo a se obter uma visão viável mais abrangente possível das situações de falha em potencial.

4 CONCLUSÃO

O desenvolvimento deste trabalho foi marcado pela escassez de conteúdo relativo ao tema da garantia da qualidade de *software* aplicado a ambientes industriais, como é o caso dos processos de beneficiamento mineral. Apesar de ser um tema já explorado em disciplinas da Engenharia da Computação e Sistemas de Informação, todos os casos de aplicação identificados durante a composição do referencial teórico se limitavam a aplicações não críticas para a segurança de pessoas, equipamentos e processos.

Ao se reconhecer a abrangência que as lógicas de programação hoje possuem no controle dos processos e se aprofundar nas questões e aspectos relevantes do tema, foi identificada uma grande oportunidade de rearranjo do planejamento, execução e controle das atividades de desenvolvimento de lógicas de programação, seja durante a implantação de projetos ou na continuidade da operação das plantas.

Tal reorganização das tarefas, pautada pela qualidade dos produtos desenvolvidos e pela confiabilidade do *software*, agrega status de função ao Analista de Testes, indivíduo dedicado não mais a validação das lógicas programadas, mas sim à identificação de erros e cenários nocivos em potencial.

Concluo, por fim, que este trabalho possui grande utilidade às equipes de implantação de projetos, engenharia e manutenção de automação no sentido de incrementar a confiabilidade das lógicas de programação de seus CLPs, tornando os processos mais seguros e previsíveis.

5 TRABALHOS FUTUROS

Em termos de aplicabilidade, o trabalho desenvolvido cria a oportunidade de expansão do protocolo de testes exposto aos demais sistemas que compõe toda a infraestrutura de automação industrial das plantas beneficiamento mineral.

Quanto à forma, há também a possibilidade de se gerar guias e procedimentos especificamente desenvolvidos para determinados sistemas e partes integrantes dos sistemas de controle, como no caso de sistemas supervisórios, malhas de controle, lógicas contendo funções de automatismo, dentre outros.

Já em relação ao método, tendo em vista uma maior produtividade e alcance na aplicação de diferentes rotinas de testes em programas de grande dimensão, destaca-se a possibilidade de exploração e desenvolvimento de técnicas automatizadas de testes e simulações.

REFERÊNCIAS

ALEXANDRE BARTIÉ. **Garantia da qualidade de *software***. Rio De Janeiro: Elsevier, 2002.

PRESSMAN, R. **Engenharia de *software***. Rio De Janeiro: Mc Graw Hill, 2002.

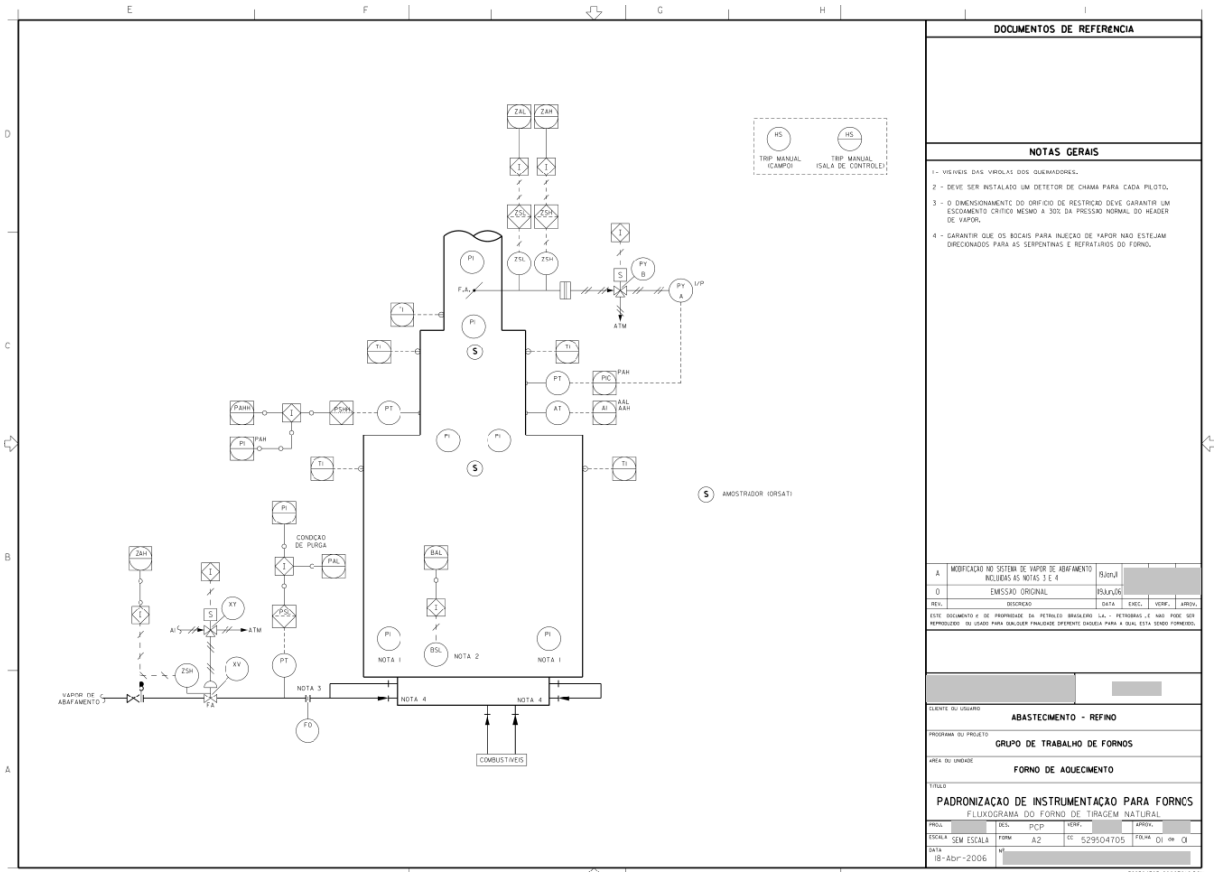
PRESSMAN, R. **Engenharia de *software***. Rio De Janeiro: Pearson, 2007.

SOMMERVILLE, I. **Engenharia de *software***. São Paulo: Pearson, 2007.

ZELKOWITZ, M. (ED.). Requirements for a Software Engineering Environment: **Proceedings of the University of Maryland Workshop.** Norwood, N.J.: Ablex Publ. Corp, 1989.

ANEXOS

ANEXO A – FLUXOGRAMA P&ID DE FORNO DE TIRAGEM NATURAL



Fonte: O autor.

ANEXO B – FORMULÁRIO DE VALIDAÇÃO DAS MODIFICAÇÕES NO SISTEMA DE CONTROLE E SUPERVISÃO

PRO-
ANEXO 1 – Formulário de validação das modificações no sistema de controle e supervisão

1- INFORMAÇÕES INICIAIS

1.1 Unidade:	
1.2 OM n°:	
1.3 PLC:	
1.4 Equipamento:	
1.5 Data:	

2 EXECUTANTE

	Nome	Matrícula	Gerência
2.1			
2.2			
2.3			
2.4			

3 ATIVIDADE REALIZADA

	Atividade	Status	Formulário
3.1	**Criação ou modificação de lógicas**	NÃO	
3.2	**Saneamento de lógicas existentes**	NÃO	
3.3	**Inserção de instrumento no sistema de controle**	SIM	Preencher o FORMULÁRIO C

PRO-
ANEXO 1 – Formulário de validação das modificações no sistema de controle e supervisão
FORMULÁRIO A - Validação de criação/modificação de lógica

3.1.1	**Aplicabilidade de Tarefa**	**Status**	**Observação**
3.1.1.1	Realizar testes de ponto a ponto para sinais digitais, analógicos ou via redes	NÃO	
3.1.1.2	Realizar verificação de variáveis quanto a duplicidade de escrita, mnemônico, descrição ou falta de variáveis associadas		
3.1.1.3	Realizar verificação do endereçamento para interface de comunicação com outros controladores ou interface com outros sistema		
3.1.1.4	Realizar verificação de consistência de programação de blocos (duplicidade, não realização de download) ou linhas de programação duplicadas		

3.1.2	**Item de Verificação**	**Status**	**Observação**
3.1.2.1	Funcionalidades da lógica conforme descritivo funcional		
3.1.2.2	Padronização da lógica		
3.1.2.3	Comentários da lógica		
3.1.2.4	Configurações de ativos de CLP		
3.1.2.5	Configurações de redes industriais		
3.1.2.6	Testes operacionais a vazio para validação das funcionalidades da lógica		
3.1.2.7	Testes operacionais em condições normais de operação para validação das funcionalidades da lógica		

3.1.1.1	**TESTE DE PONTO A PONTO PARA SINAIS DIGITAIS, ANALÓGICOS OU VIA REDES**	**Aplicabilidade**	**NA**

Item	TAG/Endereço	CLP	SUP	PIMS	IHM	Observação	Responsável

PRO
ANEXO 1 – Formulário de validação das modificações no sistema de controle e supervisão
FORMULÁRIO B - Validação de saneamento de lógica existente

3.2.1	Item de Verificação		
3.2.1.1	**Estrutura de Projeto**	**Status**	**Observação**
	Segregação de funções por subrotinas ('Functions', 'Sections', 'Subroutines'...)		
	Organização de estrutura de dados ('Estados', 'Defeitos', 'Intertravamentos', 'Ajustes',...)		
	Sequênciamento de funções na(s) subrotina(s)		
3.2.1.2	**Taxonomia (tageamento)**	**Status**	**Observação**
	Equipamentos		
	Funções		
	Itens simbólicos		
	Memórias auxiliares		
	Temporizadores		
	Contadores		
3.2.1.3	**Blocos de funções (Blocos de Válvulas; Motor / Acionamento; Analógicas; dentre outras funções específicas)**	**Status**	**Observação**
	Adequada utilização de blocos de funções existentes quanto à versão		
	Adequada utilização de blocos de funções existentes quanto à parametrização / uso dos pinos		
3.2.1.4	**Forces e lógicas inativas**	**Status**	**Observação**
	Forces ativos programados conforme procedimento vigente aplicável à localidade		
	Presença de lógicas inativas		

3.2.2 Item de validação

Item	TAG/ENDEREÇO	CLP	SUP	PIMS	IHM	Observação	Responsável
1							

PRO
ANEXO 1 – Formulário de validação das modificações no sistema de controle e supervisão
FORMULÁRIO C - Validação de inserção de instrumento no sistema de controle

3.3.1	Aplicabilidade de Tarefa	Status	Observação
3.3.1.1	Testar o sinal de leitura do instrumento na rede em unidade de engenharia		
3.3.1.2	Testar o sinal de leitura do instrumento em 4-20mA		
3.3.1.3	Testar falha de comunicação do instrumento na rede		
3.3.1.4	Testar palavra de diagnostico do instrumento		

3.3.2	Item de Verificação	Status	Observação
3.3.2.1	Verificar tags na base de dados do CLP		
3.3.2.2	Verirfcar lógica de tratamento do sinal no CLP		
3.3.2.3	Verificar se o instrumento está sendo lido na rede		
3.3.2.4	Verificar se o instrumento está com sinal de leitura no cartão de entrada		
3.3.2.5	Testar leitura inicio do range e final do range de leitura configurado		
3.3.2.6	Testar falha de leitura do instrumento no cartão de entrada analógica		
3.3.2.7	Testar os níveis de alarme associados a anológica		

3.3.1.1 TESTE DO SINAL DE LEITURA DO INSTRUMENTO NA REDE EM UNIDADE DE ENGENHARIA — **Aplicabilidade** NA

Item	TAG/ENDEREÇO	CLP	SUP	PIMS	IHM	Observação	Responsável
1							

Fonte: O autor.

ANEXO C – ROTEIRO DE TESTES DE MALHAS DE CONTROLE

LISTA DE VERIFICAÇÃO PARA CONFIGURAÇÃO					
Item	**O que**	**Como**	**Quando**	**Quem**	**OBS**
1	Verificar uma malha de controle simples	i - Verificar se SP segue a PV em manual ii - Testar se em caso de falha da PV o controlador está indo para manual e mantendo o valor da última saída antes da falha ocorrer iii - Em caso de limitação de sáida do controlador, verificar se esta limitação funciona tanto no modo manual como automático do controlador. iv - Em caso de limitação de set point, verificar se esta limitação está sendo atendida com o controlador em auto.	Durante Configuração	AEDC e UO	
2	Verificar uma malha de controle em cascata	i - Verificar os itens da malha simples para os controladores escravo e mestre ii - Verificar se de auto para cascata não ocorre bump no set point do escravo iii - Verificar se na falha da PV do mestre o escravo vai para auto ou o mestre vai para manual iv - Em caso de limitação de set point no escravo, verificar se a saída do mestre para de integrar quando essa limitação é atingida.	Durante configuração	AEDC e UO	
3	Verificar uma malha de controle em razão	i - Verificar os itens da malha simples para o controlador em razão ii - Verificar se a razão real é sempre indicada iii - Verificar se ocorre bump na transferência de automático para razão iv - Verificar se na falha da variável não controlada, o controlador degrada seu modo de operação para AUTO	Durante configuração	AEDC e UO	
4	Verificar uma malha de controle em override	i - Verificar os itens da malha simples para os controladores do override ii - Os controladores associados a restrições não devem ser configurados com o SP seguindo a PV quando no modo manual iii - Veriificar se quando um controlador perder o controle sobre a manipulada, sua ação integral é desativada e sua saída segue a saída do seletor com um bias de diferença que será o ganho do controlador vezes o erro atual	Durante configuração	AEDC e UO	
5	Verificar uma malha de controle com feedforward	i - Verificar os itens da malha simples para o controlador ii - Verificar se o FFW tem como parâmetros de sintonia ganho, lead-lag e tempo morto iii - Verificar se na falha da PV de antecipação, o seu valor fica congelado no valor da última leitura, sem balanço na saída do controlador. iv - Verificar o retorno da PV de antecipação após uma falha. Esse retorno não deve gerar balanços na saída do controlador.	Durante configuração	AEDC e UO	Anexo I
6	Verificar uma malha de controle PID GAP	i - Verificar os itens da malha simples para o controlador PID GAP ii - Veriificar se o ganho é constante em uma faixa de erro e, se após essa faixa, ele aumenta gradativamente	Durante configuração	AEDC e UO	Anexo II
7	Verificar uma malha de controle PID ON-OFF	i - Verificar os itens da malha simples para o controlador PID ON-OFF ii - Veriificar se o ganho é nulo em uma faixa de erro e, se após essa faixa, este ganho vai a um valor bem alto. Iii - Este algoritmo só funciona se o PID for incremental. Se for posicional, terá que ser feito uma lógica para a realização deste controlador.	Durante configuração	AEDC e UO	Anexo III
8	Verificar uma malha de controle em split range (2 válvulas)	i - Verificar os itens da malha simples para o controlador split range ii - Veriificar se foi configurada uma estação (manual loader) para cada válvula de forma a permitir que o operador abra separadamente essas válvulas. iii - Quando o manual loader de uma válvula estiver em manual, o outro manual loader também estará em manual iii - Quando o manual loader de uma válvula for passado para auto, o outro manual loader também será comutado para auto iv - Enquanto os manual loader estiverem em manual, o controlador de vazão deve ficar travado em manual	Durante configuração	AEDC e UO	
9	Malha de controle com seletor de sinais	i - Verificar os itens da malha simples para o controlador simples ii - Verificar se existe uma rampa no momento do chaveamento dos sinais. Essa rampa só poderá existir neste momento. iii - Outra maneira seria permitir a atuação da HS apenas com o controlador no modo manual, evitando uma descontinuidade na saída do controlador iv - No caso de falha de uma das PVs, apenas a PV selecionada deve comutar o controlador para o modo manual. Se a falha for na PV não selecionada, o controlador pode continuar em auto.	Durante configuração	AEDC e UO	
10	Malha de cálculo de média	i - Verificar se a média está sendo corretamente calculada. ii - Simular falha em um dos componentes de cálculo da média. Deverá ocorrer exclusão automática deste sinal de modo a pouco influenciar nesta média. iii 0 Simular um sinal com valor bem fora da média. Verificar se ocorre exlusão automática deste sinal no cálculo da média. A faixa que um sinal será excluído da média deve ser uma parâmetro ajustável.	Durante configuração	AEDC e UO	
11	Malha com compensação de pressão e temperatura	i - Verificar se a compensação está sendo corretamente calculada. Para isso, utilizar a planilha Compensação de Vazão ii - Simular falha da temperatura e verificar se a temperatura de projeto da placa está sendo utilizada no cálculo da correção. Também pode ser utilizado a última temperatura antes da ocorrência da falha do trasmissor de temperatura. iii - Simular falha da pressão e verificar se a pressão de projeto está sendo utilizada no cálculo da correção. Também pode ser utilizado a última pressão antes da ocorrência da falha do trasmissor de pressão. No caso de falha do PT ou TT deverá ser prevista uma HS para que o operador reabilite o cálculo da compensação a partir do transmissor de pressão ou de temperatura.	Durante configuração	AEDC e UO	
12	Iniciadores de intertravamento	Todos iniciadores de intertravamento devem ter registro de tendência no SDCD e no PI	Durante configuração	UO	-

Fonte: O autor.