UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTAÇÃO E SISTEMAS

ADRIANA FERNANDA DE OLIVEIRA

UM ESTUDO SOBRE A APLICABILIDADE DA SÉRIE DE PADRÕES ISO/IEC/IEEE 29119 EM MÉTODOS ÁGEIS DE DESENVOLVIMENTO DE *SOFTWARE*

João Monlevade
2017

ADRIANA FERNANDA DE OLIVEIRA

UM ESTUDO SOBRE A APLICABILIDADE DA SÉRIE DE PADRÕES ISO/IEC/IEEE 29119 EM MÉTODOS ÁGEIS DE DESENVOLVIMENTO DE *SOFTWARE*

Monografia apresentada ao curso Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina “Trabalho de Conclusão de Curso II”.

Orientador: Euler Horta Marinho

João Monlevade

2017

O482e Oliveira, Adriana Fernanda.
Um estudo sobre a aplicabilidade da série de padrões ISO/IEC/IEEE 29119 em métodos ágeis de desenvolvimento de software [manuscrito] / Adriana Fernanda Oliveira. - 2017.

102f.: il.: color; tabs; Quadros.

Orientador: Prof. MSc. Euler Horta Marinho.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

1. Engenharia de software. 2. Software - Testes . 3. Software - Controle de qualidade - Normas. 4. Desenvolvimento ágil de software. 5. Software - Desenvolvimento. I. Marinho, Euler Horta. II. Universidade Federal de Ouro Preto. III. Titulo.

Catalogação: ficha@sisbin.ufop.br CDU: 004.41

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

ATA DE DEFESA

Aos 24 dias do mês de agosto de 2017, às 14 horas e 00 minutos, na sala C304 do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pela aluna **Adriana Fernanda de Oliveira,** sendo a Comissão Examinadora constituída pelos professores: Prof. Me. Euler Horta Marinho, Prof. Dr. Diego Zuquim Guimarães Garcia e Prof. Dr. Fernando Bernardes de Oliveira.

A candidata apresentou a monografia intitulada: *"UM ESTUDO SOBRE A APLICABILIDADE DA SÉRIE DE PADRÕES ISO/IEC/IEEE 29119 EM MÉTODOS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE"*. A comissão examinadora deliberou, por unanimidade, pela aprovação da candidata, com nota 9,0 (Nove Vírgula Zero), concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final.

Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pela graduanda.

João Monlevade, 24 de Agosto de 2017.

Prof. Me. Euler Horta Marinho
Professor Orientador/Presidente

Prof. Dr. Diego Zuquim Guimarães Garcia
Professor Convidado

Prof. Fernando Bernardes de Oliveira
Professor Convidado

Adriana Fernanda de Oliveira
Graduanda

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

Curso de Sistemas de Informação

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

UM ESTUDO SOBRE A APLICABILIDADE DA SÉRIE DE PADRÕES ISO/IEC/IEEE 29119 EM MÉTODOS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE

Adriana Fernanda de Oliveira

Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação e aprovada pela Banca Examinadora abaixo assinada:

Prof. Me. Euler Horta Marinho
DECSI - UFOP

Prof. Dr. Diego Zuquim Guimarães Garcia
DECSI - UFOP

Prof. Dr. Fernando Bernardes de Oliveira
DECSI - UFOP

João Monlevade, 24 de Agosto de 2017

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

ANEXO III – Termo de Responsabilidade

TERMO DE RESPONSABILIDADE

Eu, Adriana Fernanda de Oliveira,
declaro que o texto do trabalho de conclusão de curso intitulado "Um Estudo Sobre a Aplicabilidade da Série de Padrões ISO/IEC/IEEE 29119 em Métodos Ágeis de Desenvolvimento de Software." é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 24 de Agosto de 2017

Adriana Fernanda de Oliveira
Assinatura do aluno

DEDICATÓRIA

Aos meus pais, meu esposo, familiares e amigos, que sempre me motivaram a chegar até aqui, e agora, como eu, sentem-se felizes por mais esta conquista em minha vida.

AGRADECIMENTOS

Primeiramente, quero agradecer a Deus, por ter me dado forças para chegar até aqui, por ter me sustentado, por ter me dado sabedoria, disciplina, e por ter me permitido conhecer pessoas tão especiais, as quais estarão para sempre em meu coração. Quero agradecer a Nossa Senhora, Virgem Maria Santíssima, por ter passado na frente de todas as minhas dificuldades e por ter me amparado com o teu manto sagrado.

A minha maravilhosa e querida mãe, Maria do Perpétuo Socorro Oliveira, por todo amor, confiança, orações e por seu exemplo de vida. Meu amor e minha gratidão pela senhora, serão eternos!

Agradeço ao meu amigo, companheiro e tão amado esposo, Fernando Araújo de Oliveira, por todo apoio, respeito e compreensão durante este longo tempo. Quero permanecer para sempre ao teu lado, conquistando sonhos e compartilhando todos os momentos da minha vida com você. Eu te amo meu amor!

A todos os meus familiares, pelo entendimento de minha ausência, pelo carinho e consideração!

Aos meus amigos da universidade, pela total colaboração, força, motivação e amizade tão importantes nesta caminhada. Em especial, a Geovana Gonçalves, Grasielle Rodrigues, Kelly Oliveira e Michel Ferreira, que não mediram esforços para me ajudar a alcançar esta vitória.

Aos mestres que realmente amam o que fazem, que possuem prazer em ensinar e transmitir o conhecimento. Principalmente, ao meu professor orientador, Euler Horta Marinho, por toda paciência, compromisso, disposição e boa vontade.

Ao Igor Muzetti Pereira, juntamente com a equipe de teste do laboratório iMobilis, por todo apoio e colaboração na realização deste trabalho.

A todas as pessoas que de algum modo contribuíram para a conclusão desta etapa tão importante em minha vida, meu muito obrigada!

“Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.” **(Charles Chaplin)**

RESUMO

Hoje em dia, as organizações estão investindo cada vez mais em desenvolvimento de *software*, porém muito mais exigente em relação à qualidade dos serviços e produtos adquiridos. O teste de *software* é de fundamental importância para garantir a qualidade do *software* desenvolvido, uma vez que possibilita a identificação de falhas antes que o produto final seja entregue ao cliente, aumentando a possibilidade de sucesso do projeto. O objetivo deste estudo é realizar uma análise sobre a aplicabilidade da série de padrões ISO/IEC/IEEE 29119 em processos ágeis de desenvolvimento de *software*, proporcionando maior eficácia nos processos de teste, a partir do atendimento dos princípios dos métodos ágeis e do foco na qualidade do *software*. Foi realizado um estudo de caso no laboratório iMobilis que contribuiu para a aplicação prática da integração dos padrões da série 29119 ao BOPE, processo de desenvolvimento de *software* que contém características comuns aos métodos ágeis.

Palavras-chave: Teste de *Software*, Melhoria do Processo de Teste, Qualidade de *Software*, Métodos Ágeis, Padrões ISO/IEC/IEEE 29119.

ABSTRACT

Nowadays, organizations are investing more and more in software development, but much more demanding in relation to the quality of the services and products acquired. The software test is of fundamental importance to guarantee the quality of the software developed, since it allows the identification of failures before the final product is delivered to the client, increasing the possibility of project success. The objective of this study is to perform an analysis on the applicability of the ISO / IEC / IEEE 29119 series of standards in agile processes of software development, providing greater efficiency in the test processes, based on the principles of agile methods and the focus on the Quality of the software. A case study was carried out in the iMobilis laboratory, which contributed to the practical application of the integration of the 29119 series standards into the BOPE, a software development process that contains characteristics common to agile methods.

Keywords: Software Testing, Test Process Improvement, Software Quality, Agile Methods, ISO / IEC / IEEE 29119 Standards

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

ISO	–	Organização Internacional de Normalização
IEC	–	Comissão Eletrotécnica Internacional
IEEE	–	Instituto de Engenheiros Eletricistas e Eletrônicos
ICEA	–	Instituto de Ciências Exatas e Aplicadas
UFOP	–	Universidade Federal de Ouro Preto
CASE	–	*Computer-aided Software Engineering*
XP	–	*Extreme Programming*
ASD	–	*Adaptive Software Development*
DSDM	–	*Dynamic Systems Development Method*
PMBOK	–	*Project Management Body of Knowledge*
ABNT	–	Associação Brasileira de Normas Técnicas
V&V	–	Verificação e Validação
STPI	_	Melhoria do Processo de Teste de *Software*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, os sistemas de *software* estão presentes em todos os lugares e têm se tornado parte integrante de cada aspecto da vida cotidiana das pessoas, sendo utilizados com diferentes finalidades, como ferramentas de trabalho, entretenimento, educação, entre outras.

Segundo SOMMERVILLE (2011), a Engenharia de *Software* é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de *software*, desde os estágios iniciais da especificação do sistema até sua manutenção. Diante desta definição, é possível compreender que a Engenharia de *Software* é crucial para garantir que o *software* atenda os padrões de qualidade, cumpra o cronograma e orçamentos inicialmente estabelecidos.

Para PRESSMAN (2011), o papel desempenhado pelo *software* tem passado por grandes mudanças ao longo dos últimos cinquenta anos. Aperfeiçoamentos significativos no desempenho do *hardware*, mudanças profundas nas arquiteturas computacionais, vasto aumento na capacidade de memória e armazenamento, e uma ampla variedade de exóticas opções de entrada e saída, tudo isso resultou em sistemas computacionais mais sofisticados e complexos.

O investimento na garantia de qualidade de *software* demonstrou ser essencial para que as organizações e as equipes envolvidas nos projetos de desenvolvimento, sejam capazes de atender, simultaneamente, a crescente demanda e a complexidade dos produtos de *software*. PEZZÉ e YOUNG (2008), afirmam que a qualidade de *software* resulta em um conjunto completo de atividades interdependentes, entre as quais os testes de *software* são necessários e estão profundamente integrados ao processo.

Para SOMMERVILLE (2011), o processo de teste de *software* visa atingir dois objetivos distintos: demonstrar que o sistema atende aos seus requisitos, e que o *software* não se comporta de maneira não correta, não desejável ou de modo diferente do que foi especificado. Assim, o teste ao qual o produto foi submetido está diretamente relacionado com a garantia de qualidade do produto final. Um teste inadequado resulta em defeitos a serem prolongados durante todo o ciclo de vida do *software*.

Muitas organizações estão adotando métodos ágeis para desenvolvimento de *software*, a fim de acompanhar as mudanças do mercado, e garantir a qualidade de *software*. Boa parte das metodologias ágeis surgiu da necessidade de descobrir maneiras melhores e diferentes de se desenvolver *software*s, promovendo um ambiente interativo e sustentável, para que novos resultados fossem obtidos. Em metodologias ágeis o teste é realizado de tal modo que, caso seja identificado algum defeito, o produto poderá retornar ao seu ambiente

de desenvolvimento para que sejam realizadas as correções necessárias, até que o mesmo esteja em total conformidade com as especificações estabelecidas para ser entregue ao cliente.

Um meio de buscar a melhoria do processo de teste, juntamente com boas práticas do gerenciamento de qualidade, é baseado na utilização de padrões propostos pela série ISO / IEC / IEEE 29119 que, de acordo com REID (2014), trata-se de um conjunto internacionalmente acordado de padrões para testes de *software* que podem ser usados em qualquer ciclo de vida de desenvolvimento de *software* e da organização. Neste contexto, está inserido o objetivo deste estudo, que é realizar uma análise sobre a aplicabilidade da série de padrões ISO / IEC / IEEE 29119 em processos ágeis de desenvolvimento de *software*. A motivação para esta integração consiste em um grande desafio, uma vez que a metodologia ágil, em sua origem, utiliza uma documentação bem limitada, enquanto a série 29119 reforça a importância de uma documentação estruturada para a melhoria do processo de teste, e consequentemente necessária para a garantia de qualidade do *software*.

1.1 Problema

Com a crescente demanda de novas tecnologias, as organizações estão investindo cada vez mais em desenvolvimento de *software*, porém muito mais exigente em relação à qualidade dos serviços e produtos adquiridos. O teste de *software* é de fundamental importância para garantir a qualidade do *software* desenvolvido, uma vez que possibilita a identificação de falhas antes que o produto final seja entregue ao cliente. A correção dessas falhas, após a implantação do *software* na organização do cliente, pode resultar em um maior custo, atrasos na resolução de problemas, riscos no projeto e insatisfação do cliente.

No entanto, mesmo existindo vários estudos comprovando a importância e a necessidade do teste de *software*, e inúmeras técnicas e metodologias de aplicações, ainda há deficiências de integração entre o processo de teste e a cultura de algumas organizações. Existe também, pouco incentivo quanto à utilização de testes, em particular, as pequenas e médias organizações não possuem recursos suficientes para adotar o uso de metodologias eficientes, e, por esta razão, possuem dificuldade para realizar processos de testes adequados, conforme apresentado por HWANG (2016).

A melhoria do processo de teste aumenta a possibilidade de sucesso do projeto, e pode ser obtida por meio da utilização da ISO/IEC/IEEE 29119 *Software Testing,* que é um conjunto de padrões para teste de *software* reconhecido internacionalmente, com início de publicação em setembro de 2013. Essas normas podem ser aplicadas em qualquer ciclo de desenvolvimento de *software* e em qualquer organização, fornecendo uma abordagem de

alta qualidade para testes que podem ser comunicados em todo o mundo. Operando de acordo com o processo proposto nas normas, a qualidade do produto pode ser garantida (AFZAL et al., 2016).

Outro ponto importante a ser considerado é o processo de desenvolvimento de *software*, que é um conjunto de atividades que auxilia na produção de *software*. Desenvolver *software* sem a utilização de nenhum processo resulta em baixa qualidade do produto final, dificultando a entrega do *software* nos prazos estabelecidos com o cliente e inviabilizando a evolução do *software*. O desenvolvimento ágil considera uma diversidade grande de abordagens referentes aos processos, uma delas é a de trabalhar com pequenas iterações com entregas ao final de cada iteração (BECK et al., 2001). Por esta razão, o teste de *software* se torna parte integrante de todo o processo de desenvolvimento.

Portanto, pretende-se enfatizar a melhoria do processo de teste nas organizações, com a finalidade de torná-lo mais eficaz, a partir do atendimento dos princípios dos métodos ágeis e o foco na qualidade do *software*. Para a condução da pesquisa, será realizado um estudo de caso no laboratório iMobilis, situado no Instituto de Ciências Exatas e Aplicadas (ICEA) da Universidade Federal de Ouro Preto, que utiliza características ágeis em seu processo de desenvolvimento de *software*, a fim de verificar como é realizado o atual processo de teste de software, identificar lacunas e propor melhorias, para que o processo de teste seja aprimorado.

1.2 Objetivos

Esta seção descreve o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivo geral

O trabalho tem como objetivo analisar se os padrões da série ISO / IEC / IEEE 29119 podem ser utilizados no processo ágil. Será realizado um estudo sobre os padrões e sobre as principais metodologias ágeis, *Extreme Programming* e *Scrum*, e, posteriormente, uma análise de como os padrões da série 29119 podem ser integrados no BOPE, atual processo de desenvolvimento de *software* do laboratório iMobilis, que possui características comuns aos métodos ágeis citados anteriormente.

Este estudo situa-se em um contexto mais amplo que é a melhoria do processo de teste nas organizações que utilizam métodos ágeis.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Estudar os conceitos relacionados ao teste de *software*: principais técnicas e critérios.
- Estudar os principais métodos ágeis.
- Estudar os padrões da série ISO/IEC/IEEE 29119.
- Fazer um levantamento, a partir da literatura, das principais abordagens para a melhoria do processo de teste nas organizações.
- Avaliar se os padrões da série ISO/IEC/IEEE 29119 pode ser integrado em métodos ágeis, levando-se em consideração as diferentes abordagens propostas para a melhoria do processo de teste.
- Analisar e identificar melhorias no processo de teste que atualmente é utilizado no laboratório iMobilis, verificando meios de aplicar os padrões da série 29119 no BOPE.

1.3 Justificativa

Mesmo sabendo da importância do processo de teste no desenvolvimento de *software*, muitas organizações ainda desenvolvem projetos sem utilizá-lo ou aplica-lo de maneira inapropriada, por diversos motivos, seja a falta de recursos, conhecimento e por vezes, até a própria cultura da empresa.

O teste deve ser aplicado durante toda a fase de desenvolvimento do *software*, para que a qualidade do produto final possa ser maximizada. O planejamento inicial do teste durante a fase de requisitos é muito importante, normalmente esclarece e refina as especificações de requisitos. Já o desenvolvimento de um plano de teste durante o projeto pode sugerir estruturas e interfaces que não apenas facilitam o teste no início do desenvolvimento, como também definem as principais interfaces com mais precisão.

Por outro lado, a qualidade também depende de cada parte do processo de *software*, ou seja, nenhuma quantidade de teste pode compensar a baixa qualidade causada por outras atividades. Daí a importância da integração dos padrões da série ISO/IEC/IEEE 29119 referente ao processo de teste com os métodos ágeis de desenvolvimento de *software*.

O laboratório iMobilis já possui um processo de desenvolvimento de *software*, denominado BOPE, que possui características comuns aos métodos ágeis, porém é

necessário avaliar todo o processo de teste utilizado atualmente e propor melhorias baseadas na integração dos padrões da série 29119.

1.4 Estrutura do trabalho

Esse trabalho está dividido em 6 seções. Na seção 2 a revisão bibliográfica é apresentada, contendo todos os conceitos necessários para o entendimento do trabalho, incluindo definições referentes ao teste de *software*, aos padrões da série ISO/IEC/IEEE 29119 e aos métodos ágeis de desenvolvimento de *software*. A seção 3 demonstra a discussão sobre as abordagens de melhoria de processos de *software*. A seção 4 apresenta a metodologia, incluindo um estudo de caso, desenvolvido no iMobilis, laboratório situado no Instituto de Ciências Exatas e Aplicadas (ICEA) da Universidade Federal de Ouro Preto. A seção 5 é composta pela apresentação e análise dos resultados. Por fim, na seção 6 são apresentadas as considerações finais do trabalho.

2 REVISÃO BIBLIOGRAFICA

Esta seção apresenta uma detalhada revisão bibliográfica, abrangendo diversos assuntos relacionados ao tema do trabalho.

2.1 Engenharia de *Software*

A Engenharia de *Software* é um termo relativamente novo, surgiu em 1968, com a necessidade de tornar o desenvolvimento de *software* confiável, fácil de manter, com desempenho satisfatório, custo e cronograma previsíveis. Não há uma definição única referente à Engenharia de *Software*, uma vez que existe uma grande diversidade de abordagens para o desenvolvimento de *software*. Contudo, princípios fundamentais de processo e de organização de sistemas estabelecem a essência de todas as definições da Engenharia de *Software*.

Segundo SOMMERVILLE (2011) a Engenharia de *Software* tem por objetivo apoiar o desenvolvimento profissional de *software* mais do que as atividades de programação individual. Trata-se de uma disciplina de engenharia onde o foco principal está em todos os aspectos da produção de *software* até a conclusão do projeto, mantendo o gerenciamento na evolução do sistema, mesmo depois de entrar em operação por parte do usuário final.

É importante ressaltar que o *software* não é apenas um programa de computador, compreende também toda a documentação, configuração e estrutura de dados necessários para a implementação do mesmo.

Para PRESSMAN (2011), a Engenharia de *Software* é uma tecnologia em camadas, conforme ilustrado na Figura 1. A pedra fundamental é o foco na qualidade, a base é constituída pelos processos, que por sua vez mantém as camadas de tecnologia coesas e possibilita o desenvolvimento de *software* de modo racional e dentro do prazo, com o auxílio das informações técnicas fornecidas pelos métodos e do suporte das ferramentas.

Figura 1 – Camadas da Engenharia de *Software*

Fonte: Engenharia de *Software* (PRESSMAN, 2011)

De fato, o desenvolvimento de um *software* envolve muitos processos complexos. No entanto, a Engenharia de *Software* oferece métodos sistemáticos para lidar com a complexidade e auxiliar na resolução dos problemas, considerando as necessidades dos clientes e os recursos disponíveis na organização.

2.2 Processo de *software*

Para se obter produtos de *software* de alta qualidade, é extremamente importante integrar boas práticas da Engenharia de *Software* com um processo de desenvolvimento eficaz. Segundo PRESSMAN (2011), processo é um conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho. O Guia PMBOK define processo como sendo um conjunto de atividades inter-relacionadas realizadas para obter um conjunto específico de produtos, resultados ou serviços (PMBOK, 2008). No contexto da Engenharia de *Software*, um processo pode ser definido como um conjunto estruturado de atividades, que visam desenvolver o produto de *software* dentro do prazo estabelecido e com qualidade.

Não existe um processo de *software* ideal, diferentes tipos de sistemas necessitam de diferentes processos de desenvolvimento, dependendo de suas especificações. De acordo com SOMMERVILLE (2011), embora existam muitos processos de *software* distintos, algumas atividades fundamentais são comuns a todos eles, como:

1. Especificação do *software*: A funcionalidade do *software* e as restrições sobre o seu funcionamento devem ser definidas.
2. Projeto e implementação de *software*: O *software* deve ser produzido para atender às especificações.
3. Validação de *software*: O *software* deve ser validado para garantir que atenda às demandas do cliente.
4. Evolução do *software*: O *software* deve evoluir para atender às necessidades de mudança dos clientes.

A maioria das organizações desenvolvem os próprios processos de desenvolvimento de *software*. No entanto, ainda existem aquelas que não fazem uso das boas práticas da Engenharia de *Software*, desenvolvendo produtos não estão em conformidade com as especificações dos usuários, aumentando consequentemente o tempo e o custo para a conclusão do projeto.

2.2.1 Modelo de processo de *software*

Para criar um produto de *software* de alta qualidade e dentro do prazo estabelecido, é necessário seguir alguns critérios, que propicie estabilidade, controle e organização das atividades.

Um modelo de processo de *software* é uma representação simplificada de um processo de *software*. Cada modelo representa uma perspectiva particular de um processo, portanto, fornece informações parciais sobre ele (SOMMERVILLE, 2011).

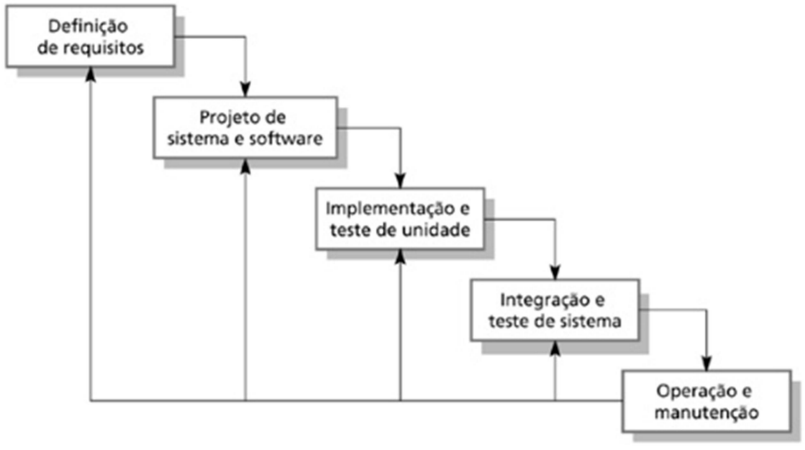
No modelo de processo de *software* deve constar atividades como: a definição das atividades para o desenvolvimento do *software*, a especificação do produto de cada atividade e indicação do papel das pessoas envolvidas.

Segundo PRESSMAN (2011), uma metodologia de processo genérica para Engenharia de *Software* estabelece cinco atividades metodológicas: comunicação, planejamento, modelagem, construção e entrega. Além disso, um conjunto de atividades de apoio é aplicado ao longo do processo, como o acompanhamento e o controle do projeto, a administração de riscos, a garantia de qualidade, o gerenciamento de configurações, as revisões técnicas e outras.

Existem diversos modelos de processo de *software* presentes na literatura, os mais utilizados nas práticas de Engenharia de *Software* são: modelo de cascata, desenvolvimento incremental e Engenharia de *Software* orientada a reuso.

Devido ao encadeamento de uma fase e outra, o modelo do processo de desenvolvimento de *software* apresentado na Figura 2, ficou conhecido como modelo de cascata ou ciclo de vida do *software*.

Figura 2 – Ciclo de Vida do *Software*



Fonte: Engenharia de *Software* (SOMMERVILLE, 2011)

O modelo de cascata considera as atividades fundamentais ao processo, compreendendo de maneira geral a especificação, o desenvolvimento, a validação e a evolução. É exemplo de um processo dirigido a planos, onde em princípio, planeja-se a programar todas as atividades do processo antes de começar a trabalhar nelas (SOMMERVILLE, 2011).

A Figura 2 representa os principais estágios do modelo em cascata. O resultado de cada estágio é a aprovação de um ou mais documentos, assinados. O estágio seguinte não deve ser iniciado até que a fase anterior seja concluída. Na prática, esses estágios se sobrepõem e alimentam uns aos outros de informações (SOMMERVILLE, 2011).

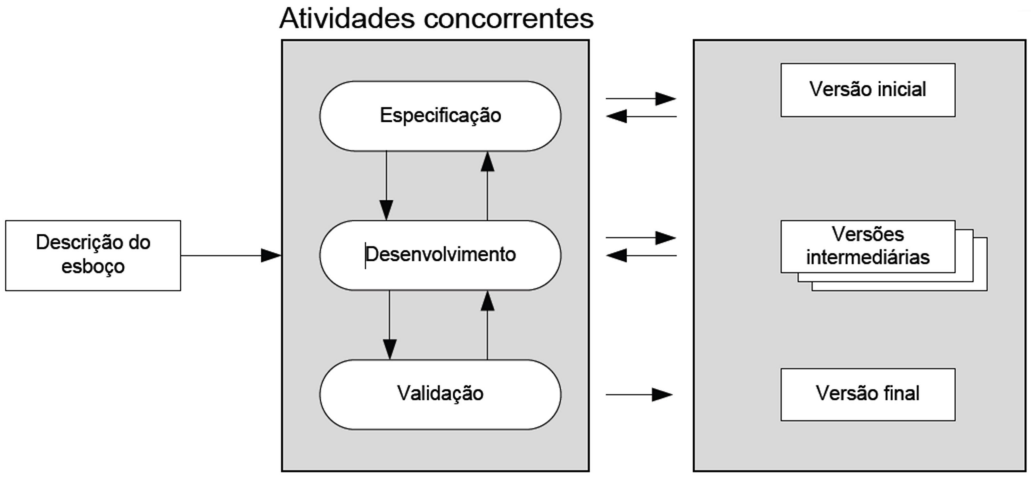
O modelo em cascata é o paradigma mais antigo da Engenharia de *Software*. Entretanto, segundo PRESSMAN (2011), alguns problemas encontrados quando se aplica o modelo têm sido objetos de questionamento de sua eficácia, como:

- Projetos reais raramente seguem o fluxo sequencial que o modelo propõe.
- Frequentemente é difícil para o cliente estabelecer explicitamente todas as necessidades. O modelo requer isso e tem dificuldade para adequar a incerteza natural que existe no início de muitos projetos.
- O cliente deve ter paciência. Uma versão operacional do programa não estará disponível antes de estar próximo do final do projeto. Um erro grave se não detectado até o programa operacional ser revisto, pode ser desastroso.

Segundo SOMMERVILLE (2011, p. 21) e PRESSMAN (2011, p. 61), o modelo em cascata deve ser utilizado somente quando os requisitos são fixos e tenham pouca probabilidade de serem alterados durante o desenvolvimento do sistema e o trabalho deve ser realizado até sua finalização de modo linear.

Outra abordagem utilizada é o modelo de processo incremental, que combina elementos dos fluxos de processos lineares e paralelos (PRESSMAN, 2011). Este modelo intercala as atividades de especificação, desenvolvimento e validação, conforme ilustrado na Figura 3. Um sistema inicial é rapidamente desenvolvido a partir de especificações abstratas, que são então refinadas com informações do cliente, para produzir um sistema que satisfaça a suas necessidades, produzindo várias versões do *software* (SOMMERVILLE, 2011). Este modelo permite iniciar o sistema pelas partes melhor compreendidas e dar *feedback* de maneira mais rápida para o cliente.

Figura 3 – Desenvolvimento Incremental

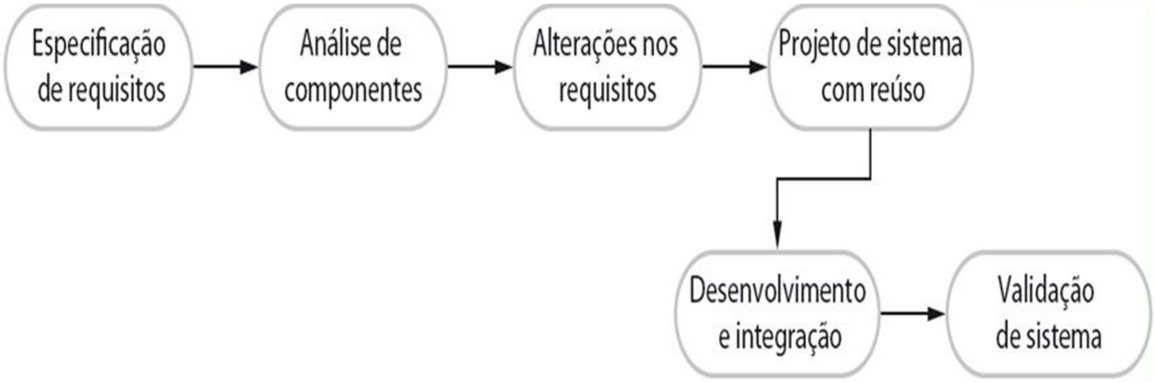


Fonte: Engenharia de *Software* (SOMMERVILLE, 2011)

O modelo de Engenharia de *Software* orientada a reuso, parte do princípio de que existem muitos componentes que podem ser reutilizados. O processo de desenvolvimento do sistema se concentra em combinar vários desses componentes em um sistema, em vez de proceder ao desenvolvimento a partir do zero, com o objetivo de reduzir o tempo de desenvolvimento (SOMMERVILLE, 2011).

O modelo genérico de processo baseado em reuso é mostrado na Figura 4 (SOMMERVILLE, 2011). Embora as etapas de especificação de requisitos e de validação sejam comparáveis com outros processos, as etapas intermediárias em um processo orientado a reuso são diferentes.

Figura 4 - Modelo genérico baseado em reuso



Fonte: Engenharia de *Software* (SOMMERVILLE, 2011).

É importante destacar que o reuso do *software* pode promover maior confiabilidade, uma vez que os componentes são testados em diferentes ambientes, reduzir a quantidade de *software* a ser desenvolvido, reduzir os riscos, já que haverá menos incertezas referentes

ao custo do desenvolvimento, e desenvolvimento em ritmo acelerado, pois vai melhorar a velocidade da produção e a entrega acontecerá mais rápida para o cliente. Porém, a gerência de versões dos componentes pode ser complexa.

As organizações tendem a integrar partes de diferentes modelos, dependendo de suas necessidades. Algumas podem optar por um modelo de processo baseado em plano mais estruturado e rigoroso, enquanto outras podem preferir um processo mais ágil e flexível.

2.2.2 Métodos Ágeis

Segundo SOMMERVILLE (2011), na década de 1980 e início da década de 1990, acreditava-se que a melhor maneira para conseguir o melhor *software* era por meio de um planejamento cuidadoso do projeto, qualidade da segurança formalizada, do uso de métodos de análise e projeto apoiado por ferramentas CASE *(Computer-aided Software Engineering)* e do processo de desenvolvimento de *software* rigoroso e controlado.

As equipes levavam muito tempo desenvolvendo um *software*, o processo era dominado pelo planejamento, projeto e documentação do sistema. SOMMERVILLE (2011) afirma que, quando esta abordagem pesada de desenvolvimento dirigido a planos é aplicada aos sistemas coorporativos de pequeno e médio porte, gasta-se mais tempo em análises de como o sistema deve ser desenvolvido do que no desenvolvimento de programas e testes.

Os métodos ágeis surgiram como uma reação aos métodos clássicos de desenvolvimento, do reconhecimento da necessidade premente de se criar uma alternativa a estes “processos pesados”, caracterizados pelo foco excessivo na criação de uma documentação completa (BECK, et al, 2001). De acordo com PRESSMAN (2011), em essência, métodos ágeis se desenvolveram em um esforço para sanar fraquezas reais e perceptíveis de Engenharia de *Software* convencional.

Em 2001, Kent Beck e outros dezesseis renomados desenvolvedores, autores e consultores da área de *software*, assinaram o “Manifesto para o Desenvolvimento Ágil de *Software*” (PRESSMAN, 2011). Estes especialistas não eram contra métodos, processos ou metodologias, e defendiam a modelagem e a documentação do *software*, mas não em excesso.

SOMMERVILLE (2011) define os métodos ágeis como sendo, universalmente, baseados em uma abordagem incremental para a especificação, o desenvolvimento e a entrega do *software*. Destinam-se a entregar *software* rapidamente aos clientes, em funcionamento, e estes podem, em seguida, propor alterações e novos requisitos a serem incluídos nas iterações posteriores do sistema. Tem como objetivo reduzir a burocracia do

processo, evitando qualquer retrabalho de valor duvidoso de longo prazo e qualquer documentação que provavelmente nunca será usada.

Por meio do manifesto ágil, passou-se a valorizar: indivíduos e interações mais que processos e ferramentas, *software* em funcionamento mais que documentação abrangente, colaboração com o cliente mais que negociação de contratos e responder a mudanças mais que seguir um plano (BECK et al., 2001).

Foram propostos doze princípios para a prática de metodologias ágeis, conforme apresentado no Quadro 1. Nem todo modelo de processo ágil aplica esses doze princípios atribuindo-lhes pesos iguais, depende da necessidade de cada projeto. Porém, os princípios demonstram o fundamento das metodologias ágeis que devem estar contidas em cada modelo.

Quadro 1: Princípios das metodologias ágeis de desenvolvimento de *software*

Princípios do Software Ágil
Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
Entregar software funcionando com freqüencia, na escala de semanas até meses, com preferência aos períodos mais curtos.
Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diáriamente, durante todo o curso do projeto.
Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
O método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
Software funcional é a medida primária de progresso.
Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis.
Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Fonte: Adaptado de Manifesto Ágil (BECK *et. al.*, 2001).

Por meio dos princípios definidos pelo manifesto ágil, criou-se várias metodologias e descrições de processos, para apoiar o desenvolvimento de *software*, como o *Extreme Programming* (XP), *Scrum*, *Adaptive Software Development* (ASD), *Dynamic Systems Development Method* (DSDM), *Crystal*, entre outros.

Conforme Somerville (2011), a maioria dos projetos de *software* inclui práticas das abordagens dirigidas a planos e ágil. Na verdade, a questão de rotular o projeto como

dirigido a planos ou ágil não é muito importante. Em última análise, a principal preocupação é desenvolver um sistema de *software* executável que atenda as necessidades do cliente e seja útil ao usuário ou organização.

2.2.2.1 *Extreme Programming*

Segundo PRESSMAN (2011), *Extreme Programming* é a abordagem mais amplamente utilizada para desenvolvimento de *software* ágil. Embora os primeiros trabalhos sobre os conceitos e métodos associado são modelo tenham ocorrido no final dos anos 1980, o trabalho seminal sobre o tema foi descrito por Kent Beck. Mais recentemente, foi proposta uma variação da XP, denominada Industrial XP (IXP). O IXP refina o modelo e visa o processo ágil especificamente para uso em grandes organizações.

De acordo com MEYER (2014), *Extreme Programming* é a abordagem ágil original, no sentido de que a sua introdução foi o evento que trouxe ideias ágeis para frente do *software* no estágio de engenharia. O modelo é menos visível hoje, grande parte do centro das atenções se mudou para *Scrum*. Mas essa mudança de moda esconde a realidade da contínua influência do método: Os princípios e práticas mais construtivos do modelo foram integrados em outras abordagens e muitos projetos os aplicam independentemente de os membros do projeto estarem ou não conscientes de sua proveniência.

A grande ideia do *Extreme Programming* é o incremento seguido da simplificação, este é o ciclo, repetido até que a equipe e o cliente estejam satisfeitos (MEYER, 2014). Duas contribuições por si só bastariam para assegurar o lugar do modelo na história da Engenharia de *Software*, segundo MEYER (2014):

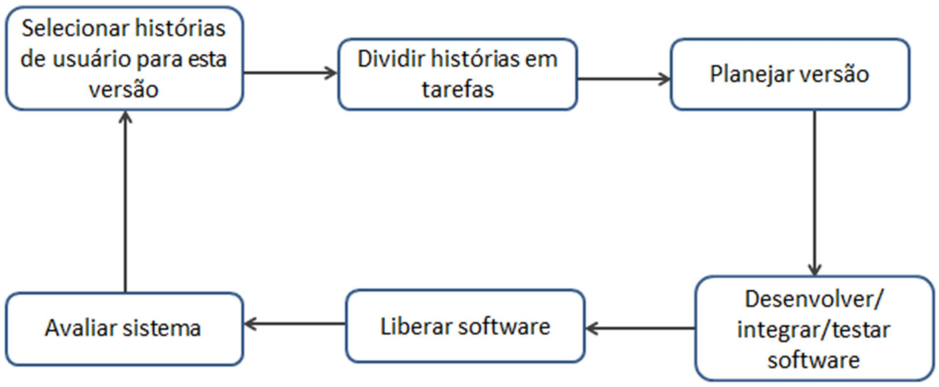
- Os projetos não devem deixar os ramos divergirem, mas integrar o código todo o tempo;
- E devem tratar os testes como um recurso, não permitindo que qualquer código seja desenvolvido sem testes.

Para WELLS (2013), o modelo XP é bem sucedido porque enfatiza a satisfação do cliente. Em vez de entregar tudo o que ele poderia desejar em algum momento do futuro, este processo entrega o *software* que ele precisa, conforme ele precisa. Permite que seus desenvolvedores respondam com confiança às mudanças nos requisitos do cliente, mesmo no final do ciclo de vida.

De acordo com SOMMERVILLE (2011), no *Extreme Programming*, os requisitos são expressos como cenários, chamados de estórias do usuário, que são implementados diretamente como uma série de tarefas. Os programadores trabalham em pares e

desenvolvem testes para cada tarefa antes da escrita do código. Todos os testes devem ser executados com sucesso quando um novo código é integrado no sistema. Há um pequeno espaço de tempo entre os releases de sistema. A Figura 5, demonstra o processo XP para produzir um incremento do sistema que está sendo desenvolvido.

Figura 5: Ciclo de um release em XP



Fonte: Adaptado de Engenharia de *Software* (SOMMERVILLE, 2011)

BECK (1999) afirma que o XP é leve, o desenvolvedor faz somente o que precisa fazer para criar valor para o cliente. É uma metodologia baseada em restrições de endereçamento no *software* desenvolvimento. Não aborda a gestão de portfólio de projetos, justificação financeira de projetos, operações, marketing ou vendas. A metodologia geralmente é interpretada como significando um conjunto de regras a seguir que garantem o sucesso. Além disso, o modelo pode trabalhar com equipes de qualquer tamanho, os valores e princípios por trás deste método são aplicáveis em qualquer escala. No entanto, as práticas precisam ser aumentadas e alteradas quando muitas pessoas estão envolvidas.

WELLS (2013) aponta que o XP melhora um projeto de *software* de cinco maneiras essenciais: comunicação, simplicidade, *feedback*, respeito e coragem. Os programadores extremos comunicam-se constantemente com seus clientes e outros programadores, mantêm seu *design* simples e limpo, recebem comentários testando seu *software* a partir do primeiro dia. Eles entregam o sistema aos clientes o mais cedo possível e implementam as mudanças conforme sugerido. Cada pequeno sucesso aprofunda seu respeito pelas contribuições únicas de cada membro da equipe. Com esta base, os programadores extremos são capazes de responder com coragem às mudanças de requisitos e tecnologia.

BECK (1999) refere-se ao modelo XP como um estilo de desenvolvimento de *software* que se concentra em excelente aplicação de técnicas de programação, comunicação clara e trabalho em equipe. Distingue-se de outras metodologias por:

- Seus ciclos de desenvolvimento curtos, resultando em cedo, concreto e contínuo comentários.
- Sua abordagem de planejamento incremental, que rapidamente vem com um plano geral que se espera que evolua durante a vida do projeto.
- Sua capacidade de agendar de modo flexível a implementação de funcionalidade, respondendo às necessidades comerciais em constante mudança.
- A dependência de testes automatizados escritos por programadores, clientes, e testadores para monitorar o progresso do desenvolvimento, para permitir o sistema evoluir e detectar os defeitos cedo.
- Sua dependência de comunicação oral, testes e código-fonte para comunicar estrutura e intenção do sistema.
- A dependência de um processo de *design* evolutivo que dura tanto quanto o sistema dura.
- A dependência da estreita colaboração de indivíduos ativamente envolvidos com talento comum.
- Sua dependência de práticas que funcionam tanto com os instintos de curto prazo os membros da equipe e dos interesses de longo prazo do projeto.

O modelo XP exige que os participantes aprendam um alto nível de técnica em serviço dos objetivos da equipe. Significa desistir de hábitos antigos de trabalhar para novos caminhos adaptados a realidade de hoje.

Conforme MEYER (2014), o XP leva princípios e práticas de senso comum a níveis extremos: “se as revisões de código forem boas, revisaremos o código o tempo todo (programação em pares); se o teste for bom, todo mundo irá testar o tempo todo (testes unitários), mesmo os clientes (teste funcional); se o *design* for bom, tornaremos parte do negócio diário de todos (refatoração)”.

O aspecto mais surpreendente do modelo XP é suas regras simples . É muito parecido com um quebra-cabeças, onde há muitas peças pequenas. Individualmente as peças não faz sentido, mas quando combinados, uma imagem completa pode ser vista. As regras podem parecer estranhas e talvez até ingênuas no início, mas são baseadas em valores e princípios sólidos (WELLS, 2013).

O modelo XP envolve um número de práticas, resumidas no Quadro 2, que atendem os princípios ágeis.

Quadro 2: Práticas do *Extreme Programming*

Princípio ou prática	Descrição
Planejamento Incremental	Os requisitos são gravados em cartões de estória e as estórias que serão incluídas em uma release são determinadas pelo tempo disponível e sua relativa prioridade. Os desenvolvedores dividem essas estórias em tarefas.
Pequenos releases	Em primeiro lugar, desenvolve-se um conjunto mínimo de funcionalidades útil, que fornece o valor do negócio. Releases do sitemas são frequentes e gradualmente adicionam funcionalidade ao primeiro release.
Projeto simples	Cada projeto é realizado para atender às necessidades atuais, e nada mais.
Desenvolvimento test-first	Um framework de testes iniciais automatizados é usado para escrever os testes para uma nova funcionalidade antes que a funcionalidade em si seja implementada.
Refatoração	Todos os desenvolvedores devem refatorar o código continuamente assim que encontrarem melhorias de código. Isso mantém o código simples e manutenível.
Programação em pares	Os desenvolvedores trabalham em pares, verificando o trabalho dos outros e prestando apoio para um bom trabalho sempre.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de modo que não se desenvolvam ilhas de expertise. Todos os conhecimentos e todos os desenvolvedores assumem responsabilidade por todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Assim que o trabalho em tarefa é concluído, ele é integrado ao sistema como um todo. Após essa integração, todos os testes de unidade do sistema devem passar.
Ritmo Sustentável	Grande quantidade de horas extras não são consideradas aceitáveis, pois o resultado final, muitas vezes, é a redução da qualidade do código e da produtividade a médio prazo.
Cliente no local	Um representante do usuário final do sistema (o cliente) deve estar disponível todo o tempo à equipe de XP. Em um processo de Extreme Programming, o cliente é um membro da equipe de desenvolvimento e é responsável por levar a ela os requisitos de sistema para implementação.

Fonte: Adaptado de Engenharia de *Software* (SOMMERVILLE, 2011)

Segundo SOMMERVILLE (2011), no processo XP os clientes estão intimamente envolvidos na especificação e priorização dos requisitos de sistemas. Os requisitos não estão especificados como listas de funções requeridas do sistema. Pelo contrário, o cliente é parte da equipe de desenvolvimento e discute cenários com os outros membros da equipe. Juntos, eles desenvolvem um cartão de estórias, englobando as necessidades do cliente. A equipe de desenvolvimento, então, tenta implementar este cenário em um release futuro do *software*.

O XP exige uma abordagem extrema para o desenvolvimento interativo. Novas versões de *software* podem ser construídas várias vezes por dia e os releases são entregues para os clientes a cada duas semanas, aproximadamente. Prazo de releases nunca são desrespeitados, se houver problemas de desenvolvimento, o cliente é consultado, e a funcionalidade é removida do release planejado (SOMMERVILLE, 2011).

Para SOMMERVILLE (2011), uma das diferenças importantes entre o desenvolvimento incremental e o desenvolvimento dirigido a planos está no modo como o sistema é testado.

Extreme Programming inclui uma abordagem de testes que reduz as chances de erros desconhecidos na versão atual do sistema. As principais características dos testes, conforme SOMMERVILLE (2011), são:

- Desenvolvimento *test-first*;
- Desenvolvimento de teste incremental a partir de cenários;

- Envolvimentos dos usuários no desenvolvimento de testes e validação;
- Uso de *frameworks* de testes automatizados.

Quando um programador constrói o sistema para criar uma nova versão, ele deve executar todos os testes automatizados existentes, bem como os testes para a nova funcionalidade. A nova construção do *software* será aceita somente se todos os testes foram executados com êxito (SOMMERVILLE, 2011).

2.2.2.2 *Scrum*

Scrum é um método de desenvolvimento ágil de *software* concebido por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 1990 (PRESSMAN, 2011).

Segundo SOMMERVILLE (2011), a abordagem *Scrum* é um método ágil geral, mas seu foco está no gerenciamento do desenvolvimento iterativo, ao invés das abordagens técnicas específicas da Engenharia de *Software* ágil. *Scrum* não prescreve o uso de práticas de programação, como programação em pares e desenvolvimento *test first*. Portanto, pode ser usado com abordagens ágeis mais técnicas, como o XP, para fornecer um *framework* de gerenciamento do projeto.

MEYER (2014) define que, a ideia principal do *Scrum* é o congelamento de requisitos durante iterações de curtas. Trata-se de um dos principais desafios da Engenharia de *Software*: como lidar com a mudança. *Scrum*, em particular, transformou a ideia geral de iterações de desenvolvimento em uma disciplina precisa, com regras que codificam os objetivos, duração e gerenciamento de iterações individuais. O modelo de iteração resultante, o *sprint*, é rápido tornando-se o padrão da indústria, além das equipes que aplicam explicitamente o *Scrum*.

Segundo PRESSMAN (2011), o *Scrum* engloba um conjunto de padrões de processos enfatizando prioridades do projeto, unidades de trabalho compartimentalizados, comunicação e *feedback* frequente por parte dos clientes. Estes padrões provaram ser eficazes para projetos com prazos de entregas apertados, requisitos mutáveis e críticos de negócio.

No *Scrum* existem três fases. A primeira é uma fase de planejamento geral, em que se estabelecem os objetivos gerais do projeto e da arquitetura do *software*. Em seguida, ocorre uma série de ciclos de *sprint*, sendo que cada ciclo desenvolve um incremento do sistema. Finalmente, a última fase do projeto encerra o projeto, completa a documentação exigida, como quadros de ajuda do sistema e manuais do usuário, e avalia as lições aprendidas com o projeto (SOMMERVILLE, 2011).

A característica inovadora do *Scrum* é sua fase central, chamada ciclos de *sprint*. Um *sprint* do *Scrum* é uma unidade de planejamento na qual o trabalho a ser feito é avaliado, os recursos para o desenvolvimento são selecionados e o *software* é implementado. As principais características desse processo, segundo SOMMERVILLE (2011), são:

1. *Sprints* são de duração fixa, normalmente de duas a quatro semanas.
2. O ponto de partida para o planejamento é o *backlog* do produto, que é a lista do trabalho a ser feito no projeto. Durante a fase de avaliação do *sprint*, este é revisto, e as prioridades e os riscos são identificados. O cliente está intimamente envolvido nesse processo e, no início de cada *sprint*, pode introduzir novos requisitos ou tarefas.
3. A fase de seleção envolve todos da equipe do projeto que trabalham com o cliente para selecionar os recursos e a funcionalidade a ser desenvolvida durante o *sprint*.
4. Uma vez que todos estejam de acordo, a equipe se organiza para desenvolver o *software*. Reuniões diárias rápidas, envolvendo todos os membros da equipe, são realizadas para analisar os progressos e, se necessário, reavaliar as prioridades. Nesta etapa a equipe está isolada do cliente e da organização, com todas as comunicações canalizadas por meio do chamado *Scrum Master*, que visa proteger a equipe de desenvolvimento de distrações externas.
5. No fim do *sprint*, o trabalho é revisto e apresentado aos *stakeholders*. O próximo ciclo do *sprint* começa em seguida.

As urgências *sprint*s consistem de unidade de trabalho solicitadas para atingir um requisito estabelecido no *backlog* e que precisa ser ajustado dentro de um prazo já fechado (janela de tempo), tipicamente 30 dias. As reuniões *Scrum*, duram aproximadamente 15 minutos, realizadas diariamente. São feitas perguntas referentes ao que foi realizado desde a última reunião, quais obstáculos encontrados e o que se planeja realizar até a próxima reunião (PRESSMAN, 2011)

2.3 Qualidade de *software*

Segundo PRESSMAN (2011), o termo qualidade de *software*, pode ser definido como sendo uma gestão de qualidade efetiva aplicada de modo a criar a um produto útil que forneça valor mensurável para aqueles que produzem e para aqueles que o utilizam.

A ISO/IEC 25010, padrão internacional que estabelece a qualidade de produto de *software*, identifica oito características como sendo fundamentais a qualidade de *software*, são elas: adequação funcional, eficiência de desempenho, compatibilidade, usabilidade, confiabilidade, segurança, manutenção e portabilidade. Em conjunto, estas características formam o modelo de qualidade do produto, conforme apresentado na Figura 6. É importante

que as características de qualidade sejam especificadas, medidas e avaliadas sempre que possível usando métodos de medição validados ou amplamente aceitos.

Figura 6: Modelo de qualidade externa e interna da ISO / IEC 25010

Fonte: ISO / IEC 25010 (2011)

Para PEZZÉ e YOUNG (2008), o processo da qualidade deve ser estruturado visando à completude, a oportunidade e a relação custo benefício. Completude significa que atividades apropriadas são planejadas para detectar cada classe importante de falhas, que são dependentes do domínio da aplicação, da organização e das tecnologias envolvidas. Oportunidade refere-se que as falhas são detectadas em tempo hábil, o que na prática sugere que são detectadas tão cedo quanto possível. Já a relação custo benefício implica que, dadas as restrições de completude e oportunidade, escolhe-se as atividades conforme seu custo e sua eficiência.

PEZZÉ e YOUNG (2008), afirmam ainda que, a qualidade de *software* resulta em um conjunto completo de atividades interdependentes, entre as quais os testes são necessários e estão profundamente integrados ao processo. As atividades de testes devem ocorrer ao longo do desenvolvimento e da evolução do sistema de *software*, desde o início da engenharia de requisitos até a entrega e subsequente evolução.

2.4 Teste de *software*

O desenvolvimento de um *software* não é uma tarefa simples, dependendo das características do sistema a ser criado, pode-se tornar uma atividade muito complexa. Diante desta realidade, muitos problemas podem surgir e culminar em um produto que não atenda as especificações do cliente. Para que estes problemas não sejam prolongados no ciclo de vida do *software*, ou seja, para que os erros sejam descobertos antes do *software* ser liberado para utilização, é fundamental integrar processos de testes no desenvolvimento do *software*, aumentando a qualidade do produto final.

Segundo SOMMERVILLE (2011), o processo de teste tem dois objetivos distintos:

1. Demonstrar ao desenvolvedor e ao cliente que o *software* atende a seus requisitos.
2. Descobrir situações em que o *software* se comporta de maneira incorreta, indesejável, ou diferente das especificações.

SOMMERVILLE (2011) afirma ainda que o primeiro objetivo leva a testes de validação, nos quais espera-se que o sistema execute corretamente usando determinado conjunto de casos de testes que refletem o uso esperado do sistema. O segundo objetivo leva a testes de defeitos, nos quais os casos de testes são projetados para expor os defeitos.

É importante destacar que defeitos, erros e falhas possuem definições distintas, sendo necessário conhecê-las para compreender o teste de *software*. O defeito é uma instrução incorreta ou inexistente, que resulta em um erro. O erro refere-se a um estado inconsistente ou inesperado, sendo uma consequência do defeito, que pode ou não causar uma falha, dependendo de sua gravidade. A falha por sua vez, se caracteriza por um desvio da especificação e ocorre em virtude da existência dos erros. Ou seja, o defeito gera o erro, que causa a falha.

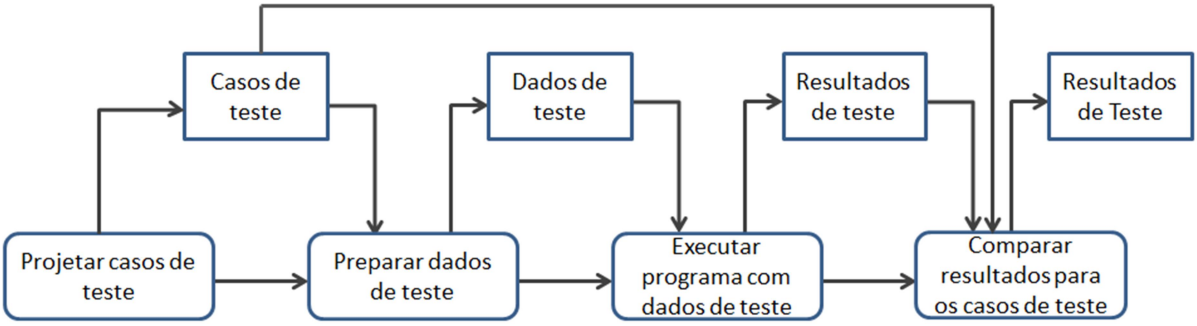
O teste é parte de um amplo processo de verificação e validação (V&V). Conforme PRESSMAN (2011), a verificação refere-se ao conjunto de tarefas que garantem que o *software* implementa corretamente uma função específica. Já a validação, refere-se a um conjunto de tarefas que asseguram que o *software* foi criado e pode ser rastreado segundo os requisitos do cliente. As atividades V&V podem e devem ser conduzidas durante todo o processo de desenvolvimento de *software*, desde a definição de requisitos até o produto final.

Identificar e revelar o máximo de falhas quanto possível, é objetivo de qualquer processo de teste de *software*. Por este motivo, sua característica é ser destrutivo e não construtivo. Porém, o fato de concluir o teste não garante que o *software* não possui mais erros, a finalização demonstra que o produto atingiu um nível desejado de funcionalidade e qualidade.

A Figura 7 representa um modelo abstrato do processo tradicional de testes. Os casos de testes são especificações das entradas para o teste e da saída esperada do sistema, além de uma declaração do que está sendo testado. Os dados do teste são as entradas criadas para testar um sistema. Às vezes, os dados de teste podem ser gerados automaticamente, mas a geração automática de casos de testes é impossível, pois as pessoas que entendem o propósito do sistema devem ser envolvidas para especificar os resultados esperados. No entanto, a execução do teste pode ser automatizada, os

resultados esperados são automaticamente comparados aos resultados previstos, por isso não há necessidade de uma pessoa para procurar erros e anomalias na execução dos testes (SOMMERVILLE, 2011).

Figura 7: Modelo de processo de teste de *software*



Fonte: Adaptado de Engenharia de *Software* (SOMMERVILLE, 2011)

PEZZÉ e YOUNG (2008), reconhecem que o melhor indicador para prever o custo de corrigir um defeito de *software* é o tempo entre sua introdução e sua detecção. Um defeito introduzido durante a engenharia de requisitos é relativamente barato de corrigir neste estágio, mas pode se tornar extremamente caro se só for revelado por uma discussão de resultados de teste de aceitação ou por um usuário de sistema em campo.

De acordo com PRESSMAN (2011), o teste muitas vezes requer mais trabalho de projeto do que qualquer outra ação da Engenharia de *Software*. Se for feito casualmente, perde-se tempo, fazem-se esboços desnecessários, e, ainda pior, erros passam sem ser detectados.

Segundo SOMMERVILLE (2011), o sistema de *software* tem de passar por três estágios de teste:

- Testes em desenvolvimento: O sistema é testado durante o desenvolvimento para descobrir boas e defeitos.
- Testes de release: Uma equipe de teste independente testa uma versão completa do sistema antes que ele seja liberado para os usuários.
- Teste de usuário: Os usuários ou potenciais usuários de um sistema testam o sistema em seu próprio ambiente.

Durante o desenvolvimento, o teste é dividido em fases com objetivos distintos. No geral, são estabelecidas três fases: teste de unidade, teste de integração ou componentes e teste de sistemas. As etapas envolvidas na realização detestes, concentram-se inicialmente em testar um único componente ou um pequeno grupo de componentes relacionados, para descobrir erros nos dados e na lógica do processamento que foram encapsulados pelos

componentes. Posteriormente, os componentes são integrados até que o sistema completo esteja pronto.

2.4.1 Teste de unidade

De acordo com DELAMARO et al. (2007), o teste de unidade tem como foco as menores unidades de um programa, que podem ser funções, procedimentos, métodos ou classes. Espera-se que sejam identificados erros relacionados a algoritmos incorretos ou mal implementados, estrutura de dados incorreta ou simples erros de programação.

Ao testar as classes de objeto, deve-se projetar os testes para fornecer uma cobertura de todas as características do objeto. Isso implica que se deve testar todas as operações associadas ao objeto, definir e verificar o valor de todos os atributos associados ao objeto e colocar o objeto em todos os estados possíveis (SOMMERVILLE, 2011).

PRESSMAN (2011) representa o teste de unidade como ilustrado na Figura 8. Para ele, a interface do módulo é testada para assegurar que as informações fluam corretamente para dentro e para fora da unidade de programa que está sendo testada.

Figura 8: Teste de Unidade

Fonte: Engenharia de *Software* (PRESSMAN, 2011)

A estrutura de dado local é examinada para garantir que os dados armazenados temporariamente mantenham sua integridade durante todos os passos na execução de um algoritmo. Todos os caminhos independentes da estrutura de controle são usados para assegurar que todas as instruções em um módulo tenham sido executadas pelo menos uma vez. As condições limites são testadas para garantir que o módulo opere adequadamente

nas fronteiras estabelecidas para limitar ou restringir o processamento. Finalmente, são testados todos os caminhos de manipulação de erro (PRESSMAN, 2011).

Casos de teste devem mostrar que, quando usado como esperado, o componente sendo testado, faz o que supostamente ele deveria fazer. Uma boa prática é projetar casos de testes de unidade antes de desenvolver o código para um componente. Desta maneira, é possível assegurar que o código desenvolvido passará nos testes.

2.4.2 Teste de integração ou componente

No teste de integração, que deve ser realizado após serem testadas as unidades individualmente, a ênfase é dada na construção da estrutura do sistema. À medida que as diversas partes do *software* são colocadas para trabalhar juntas, é preciso verificar se a interação entre elas funciona de maneira adequada e não leva a erros (DELAMARO et al., 2007).

O teste de integração é uma técnica sistemática para construir a arquitetura de *software* ao mesmo tempo em que conduz testes para descobrir erros associados com as interfaces. O objetivo é construir uma estrutura de programa determinada pelo projeto a partir de componentes testados em unidade (PRESSMAN, 2011).

Os casos de testes não são aplicados aos componentes individuais, mas sim à interface de componente criada pela combinação desses componentes. Cada vez que um novo módulo é acrescentado como parte do teste de integração, o *software* muda e novos caminhos de fluxo de dados são estabelecidos. Neste contexto, a reexecução do mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados, é conhecido como teste de regressão (PRESSMAN, 2011).

Segundo PRESSMAN (2011), à medida que o teste de integração progride, o número de testes de regressão pode crescer muito. Portanto, o conjunto de testes de regressão deve ser projetado de modo a incluir somente aqueles testes que tratam de uma ou mais classes de erros em cada uma das funções principais do programa. É impraticável e ineficiente reexecutar todos os testes para todas as funções do programa quando ocorre uma alteração.

2.4.3 Teste de sistema

Depois que se tem o sistema completo, com todas as suas partes integradas, inicia-se o teste de sistema. O objetivo é verificar se as funcionalidades especificadas nos

documentos de requisitos estão todas corretamente implementadas (DELAMARO et al., 2007).

Segundo SOMMERVILLE (2011), o teste de sistema certifica se os componentes são compatíveis, se interagem corretamente e transferem os dados no momento certo, por suas interfaces. Afirma ainda que, certamente, sobrepõem-se ao teste de componente, mas existem duas diferenças importantes:

- Durante o teste de sistema, os componentes reusáveis que tenham sido desenvolvidos separadamente e os sistemas de prateleira podem ser integrados com componentes recém desenvolvidos. Então, o sistema completo é usado.
- Nesse estágio, componentes desenvolvidos por diferentes membros da equipe ou grupos podem ser integrados. O teste de sistema é um processo coletivo, não individual.

De acordo com PRESSMAN (2011), teste de sistema é na realidade uma série de diferentes testes cuja finalidade primária é exercitar totalmente o sistema. Existem por exemplo testes de recuperação, testes de segurança, testes de desempenho, teste de disponibilização, entre outros. Embora cada um destes testes tenha finalidades distintas, todos funcionam no sentido de verificar se os elementos do sistema foram integrados adequadamente e executam as funções a eles alocadas.

2.5 Padrões de Teste ISO / IEC / IEEE 29119

A ISO (Organização Internacional de Normalização) e a IEC *(International Electrotechnical Comission*) constituem o sistema especializado de normalização a nível mundial. Os organismos nacionais que são a ISO ou IEC participam no desenvolvimento de Normas Internacionais por meio de comitês técnicos estabelecidos pela respectiva organização para lidar com domínios específicos da atividade técnica. No campo da tecnologia da informação, a ISO e IEC estabeleceram um comitê técnico conjunto, ISO / IEC JTC 1, cuja principal tarefa é preparar Normas Internacionais (ISO / IEC / IEEE 29119-1, 2013).

Os documentos dos Padrões IEEE são desenvolvidos dentro das Sociedades IEEE e os Comitês da IEEE *Standards Association* (IEEE-SA) *Standards Board*. O IEEE desenvolve seus padrões a partir de um processo de desenvolvimento de consenso, aprovado pelo *American National Standards Institute*, que reúne voluntários que representam pontos de vista variados e interesses para alcançar o produto final. O IEEE administra o processo e estabelece regras para promover a equidade no processo de desenvolvimento do consenso,

mas não consiste em avaliar, testar ou verificar independentemente a exatidão de qualquer informação contida em seus padrões (ISO / IEC / IEEE 29119-1, 2013).

ISO / IEC / IEEE 29119 é um conjunto internacionalmente acordado de padrões para testes de *software* que podem ser usados em qualquer ciclo de vida ou organização de desenvolvimento de *software*. Fornece as organizações uma abordagem de alta qualidade para testes que podem ser comunicados em todo o mundo.

Existem muitos tipos diferentes de *software*, organizações e metodologias. Os domínios de *software* incluem a tecnologia da informação, os computadores pessoais, embutidos, móveis e muitas outras classificações. As organizações de *software* variam de pequeno a grande porte, localizadas em diversas regiões, e disponibilizando serviços com várias finalidades. As metodologias de *software* podem incluir a orientações de objeto, tradicional, orientada a dados e ágil. Estes e outros fatores influenciam o teste de *software*. Esta série de Padrões internacionais podem suportar testes em muitos contextos diferentes.

Existem atualmente cinco padrões que compõe a série ISO / IEC / IEEE 29119. Inicialmente, os conceitos gerais de testes de *software* são discutidos. O papel do teste de *software* em uma organização e o contexto do projeto é descrito. O teste de *software* em um ciclo genérico de *software* é explicado, descrevendo como o teste de *software* se encaixa em diferentes modelos de ciclo de vida. O uso de diferentes práticas no planejamento de testes é apresentado e como a automação pode ser utilizada para suportar testes.

É impossível testar um sistema de *software* exaustivamente, portanto, o teste é uma atividade de amostragem. Uma variedade de práticas, técnicas e tipos de testes existem para ajudar na escolha de uma amostra adequada. Uma premissa chave deste padrão é a ideia de realizar testes ótimos dentro das limitações dadas e usando uma abordagem baseada em risco.

Os riscos podem ser categorizados de várias maneiras. Por exemplo, os riscos podem ser identificados em termos que não satisfaçam requisitos específicos ou coletivos, deixando de cumprir as obrigações contratuais, relacionadas com o progresso malsucedido ou que um produto de trabalho não atinja o seu comportamento esperado. Ao realizar o teste baseado em risco, a análise de risco é usada para identificar e marcar os riscos, para que a percepção riscos no sistema entregue, possa ser identificada, priorizada, categorizada e tratada.

2.5.1 ISO / IEC / IEEE 29119-1: Conceitos e definições

O objetivo da ISO / IEC / IEEE 29119-1 é facilitar a compreensão e o uso de todos os outros padrões na série 29119. ISO / IEC / IEEE 29119-1 introduz o vocabulário em que

todos os padrões da série 29119 são construídos e fornece exemplos da aplicação de cada conceito na prática (ISO / IEC / IEEE 29119-1, 2013). A Parte 1 é informativa e fornece definições, uma descrição dos conceitos de testes de *software* e meios de aplicar os processos, documentos e técnicas definidos na série 29119.

Inicialmente, os conceitos gerais de teste de *software* são discutidos. O papel do teste de *software* em um ambiente organizacional e o contexto do projeto é descrito. Os testes de *software* em um ciclo de vida genérico de *software* são explicados, processos de teste de *software* e subprocessos podem ser estabelecidos para itens de teste específicos ou com objetivos. Ele descreve como o teste de *software* se encaixa em diferentes modelos de ciclo de vida. O uso de diferentes práticas no planejamento do teste é apresentado, bem como o modo de como a automação pode ser usada para suportar testes.

2.5.1.1 Conceitos de introdução ao teste de *software*

Segundo a ISO / IEC / IEEE 29119-1 (2013), os principais objetivos do teste de *software* são: fornecer informações sobre a qualidade do item de teste e qualquer risco residual em relação a quanto o item de teste foi testado, encontrar defeitos no item de teste antes de sua liberação para uso e mitigar os riscos para as partes interessadas da qualidade do produto.

O processo de teste organizacional define e mantém as políticas de teste e estratégias de teste que se aplicam a projetos e funções da organização. O teste deve ser planejado, monitorado e controlado.

O teste pode ser estático ou dinâmico. Teste estático é um tipo de teste no qual um item de teste é examinado contra um conjunto de qualidade ou outros critérios, podendo incluir o uso de ferramentas que encontram defeitos no código ou documentos, sem a execução do código. Para testes estáticos esta norma reconhece e identifica o papel do testador nestas atividades, embora possam ser realizados por outros grupos no âmbito de um projeto. Já o teste dinâmico consiste em mais do que apenas verificar itens de teste executáveis, ele também inclui atividades de preparação e atividades de acompanhamento (ISO / IEC / IEEE 29119-1, 2013).

Verificação é a confirmação, por meio do fornecimento de evidência objetiva, de que os requisitos especificados foram cumpridos em um determinado item de trabalho. Validação demonstra que o item de trabalho pode ser usado pelos usuários para suas tarefas específicas. Teste, seja estático ou dinâmico, deve ter como objetivo proporcionar ambos os tipos de confirmação, embora com a expectativa de que a confirmação não será imediata por causa da descoberta dos defeitos (ISO / IEC / IEEE 29119-1, 2013).

Devido à complexidade do *software*, não é possível testar exaustivamente todos os aspectos de qualquer determinado item de teste. Testadores devem reconhecer que testes exaustivos não são possíveis e que as atividades de teste devem ser direcionadas para melhor cumprir os objetivos do teste para um item de teste.

2.5.1.2 Teste de *software* em um contexto organizacional e de projeto

As empresas envolvidas no desenvolvimento ou aquisição de produtos de *software* têm interesse no desenvolvimento, usando processos eficazes, eficientes e repetíveis. Para conseguir isso, eles geralmente desenvolvem um conjunto robusto de processos de ciclo de vida de *software*, que são aplicadas aos projetos de desenvolvimento que eles realizam. O padrão ISO / IEC / IEEE 29119-1 pretende ser útil tanto para adoção em toda a organização e para uso em projetos específicos. A organização ao adotar o padrão, pode ainda completá-lo com procedimentos adicionais, práticas, ferramentas e políticas, conforme necessário.

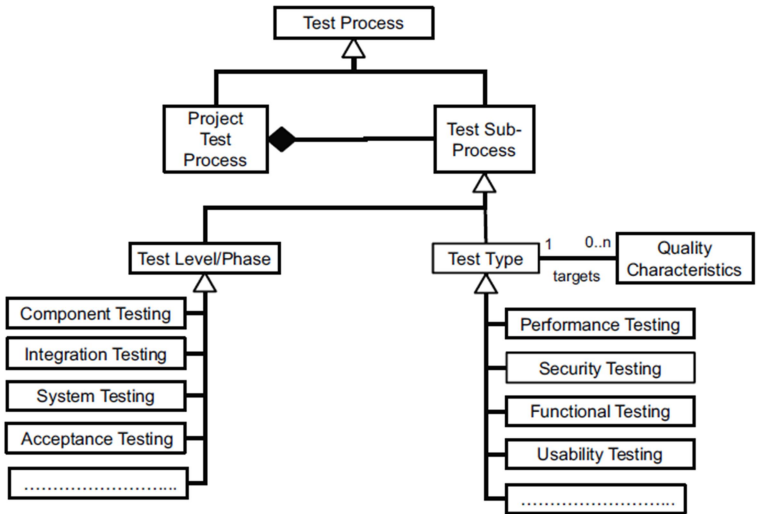
Segundo a ISO / IEC / IEEE 29119-1 (2013), o plano geral do projeto deve incluir a consideração das atividades de teste a serem realizadas como parte do projeto. Um plano de teste do projeto deve refletir tanto a Política de Teste Organizacional, a Estratégia de Teste Organizacional e os desvios dessas diretrizes organizacionais. Ele também deve explicar as restrições dadas no plano geral do projeto. Um elemento importante do planejamento de teste é a pesagem das diversas necessidades de teste e o equilíbrio de recursos entre os vários testes.

O plano de teste do projeto descreve a estratégia global para testes e o processo de teste a ser usado. Isto estabelece o contexto de testes para o projeto a partir da determinação dos objetivos, práticas, recursos e cronograma. Também identifica subprocessos de teste aplicáveis, como por exemplo, testes de sistema e testes de desempenho. Os subprocessos identificados são, em seguida, descritos no plano de ensaio.

Cada plano de ensaio do subprocesso pode tratar mais do que um nível de teste e pode dirigir mais de um tipo de teste. O plano de teste do subprocesso também descreverá a estratégia para a execução do teste.

A Figura 9 mostra a implementação dos processos de testes, em particular os níveis de teste e os tipos de testes. Cada nível de teste é uma aplicação específica do subprocesso de teste genérico (por exemplo, fase de teste de componentes, o nível de teste de aceitação). Cada tipo de teste é uma aplicação específica do subprocesso de teste genérico (por exemplo, testes de desempenho, testes de usabilidade). O diagrama também ilustra a relação entre os tipos de teste e as características de qualidade.

Figura 9: A relação entre o subprocesso de ensaio genérico, os níveis e tipos de teste



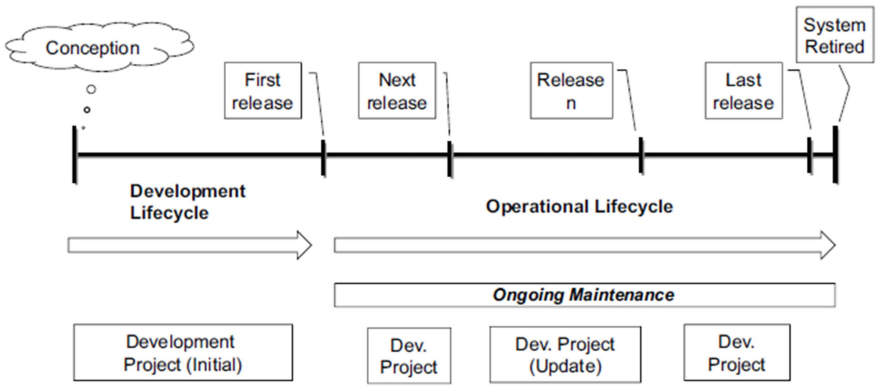
Fonte: ISO / IEC / IEEE 29119-1 (2013)

Em geral, um projeto de teste é dividido em subprocessos (por exemplo, teste de componente, teste de sistema), e estes também devem ser geridos, executados, e relatados de uma maneira semelhante ao projeto global de teste. Os processos de gerenciamento de teste também pode ser aplicados para testar subprocessos.

2.5.1.3 Processo de teste no ciclo de vida do *software*

Um ciclo de vida do *software* compreende geralmente vários estágios. A Figura 10 mostra que este ciclo muitas vezes é composto de um ou mais ciclos de desenvolvimento.

Figura 10: Ciclo de vida do *software*



Fonte: ISO / IEC / IEEE 29119-1 (2013)

O *software* tem um ciclo de vida esperado desde a sua concepção inicial até a sua retirada de produção. É a estrutura contendo processos, atividades e tarefas envolvidas no

desenvolvimento, operação e manutenção de um produto de *software*. Um ciclo de vida de desenvolvimento é gerido e controlado em um projeto de desenvolvimento.

Em cada um dos subprocessos de desenvolvimento algo é produzido, pode ser um documento altamente estruturado e detalhado ou pode ser informalmente documentado. Cada produto de trabalho, componente do sistema de *software*, e o sistema completo de *software* é um potencial item de teste.

O processo de manutenção deve ocorrer em nível aceitável de confiabilidade e disponibilidade do sistema. Isso envolve a produção de novas versões do sistema com correções de maiores defeitos encontrados na operação e, possivelmente, a introdução de alterações de alta prioridade para a funcionalidade.

Dentro de uma organização, processos de suporte são necessários para ajudar o ciclo de vida de desenvolvimento de *software*. Alguns destes são: garantia da qualidade, gerenciamento de projetos, gerenciamento de configurações e melhoria de processos (ISO / IEC / IEEE 29119-1, 2013).

A garantia de qualidade refere-se a um conjunto de processos e atividades de apoio planejadas e necessárias para proporcionar confiança adequada de que um processo ou produto de trabalho, cumpre requisitos técnicos ou de qualidade estabelecidos. Isto é conseguido por meio de imposição de métodos, padrões, ferramentas e habilidades que são reconhecidos como prática adequada para o contexto. O processo de garantia de qualidade utiliza os resultados do teste e outras informações para investigar, classificar e relatar qualquer problema na concepção, planejamento ou execução dos processos de Engenharia de *Software* (ISO / IEC / IEEE 29119-1, 2013).

O gerenciamento de projeto refere-se aos processos de suporte que são utilizados para planejar e controlar o curso de um projeto, incluindo a gestão do projeto de teste dentro do projeto global (ISO / IEC / IEEE 29119-1, 2013). A estimativa, análise de risco, e programação das atividades de teste devem ser consolidadas com o planejamento geral do projeto. O plano do projeto, que é um item de informação do processo de gerenciamento de projeto, é portanto, uma entrada para o processo de gerenciamento de testes.

O propósito de gerenciamento de configuração é estabelecer e manter a integridade dos produtos de trabalho. É uma boa prática para testar um sistema de gerenciamento de configuração do projeto antes da sua utilização operacional da organização ou, para saber se ele atende aos requisitos organizacionais ou de projeto.

A melhoria de processos de teste considera ações que buscam alinhar os processos aos objetivos de uma organização, de modo mais eficaz e eficiente. Os processos de teste e o processo de melhoria de processos interagem de duas maneiras, os processos de teste

fornecem informações sobre quais ações de melhoria de processo pode ser devem ser realizadas, e os processos de teste em si podem ser objeto de melhoria de processos.

2.5.2 ISO / IEC / IEEE 29119-2: Processos de teste

O objetivo da ISO / IEC / IEEE 29119-2 é definir um modelo de processo genérico para teste de *software* que possa ser usado em qualquer ciclo de desenvolvimento de *software*. O modelo especifica processos de teste que podem ser usados para governar, gerenciar e implementar testes de *software* em qualquer organização, projeto ou atividade de teste (ISO / IEC / IEEE 29119-2, 2013).

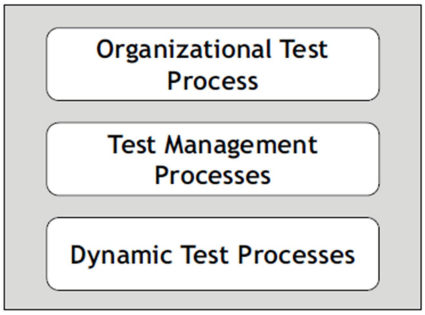
A ISO / IEC / IEEE 29119 suporta testes dinâmicos, testes funcionais e não funcionais, manual e testes automatizados, e testes com *script* e sem *script*. Cada processo é definido usando o modelo de processo genérico fornecido nas Diretrizes ISO / IEC TR 24774: 2010. Os testes são uma abordagem chave para a mitigação de riscos no desenvolvimento de *software*. Este padrão segue uma abordagem de teste baseada em risco. O teste baseado em risco é uma abordagem de melhores práticas para a estratégia e o gerenciamento de testes, pois permite que o teste seja priorizado e focado nas características e atributos de qualidade mais importantes de cada sistema em teste.

Esta parte do ISO / IEC / IEEE 29119 destina-se, entre outros, a testadores, gerentes de teste, desenvolvedores e gestores de projetos, particularmente os responsáveis pelo governo, gerenciamento e implementação de testes de *software*. A implementação deste padrão normalmente envolve a seleção de um conjunto de processos adequados para a organização ou projeto (ISO / IEC / IEEE 29119-2, 2013).

2.5.2.1 Modelo multicamadas do processo de teste

Este padrão agrupa as atividades de teste que podem ser realizadas durante o ciclo de vida de um sistema de *software*.

Figura 11: Modelo multicamadas do processo de teste



Fonte: ISO / IEC / IEEE 29119-2 (2013)

Como mostrado na Figura 11, o processo de teste é baseado em um modelo de processo de três camadas, abrangendo o teste organizacional, o gerenciamento de teste e o teste dinâmico.

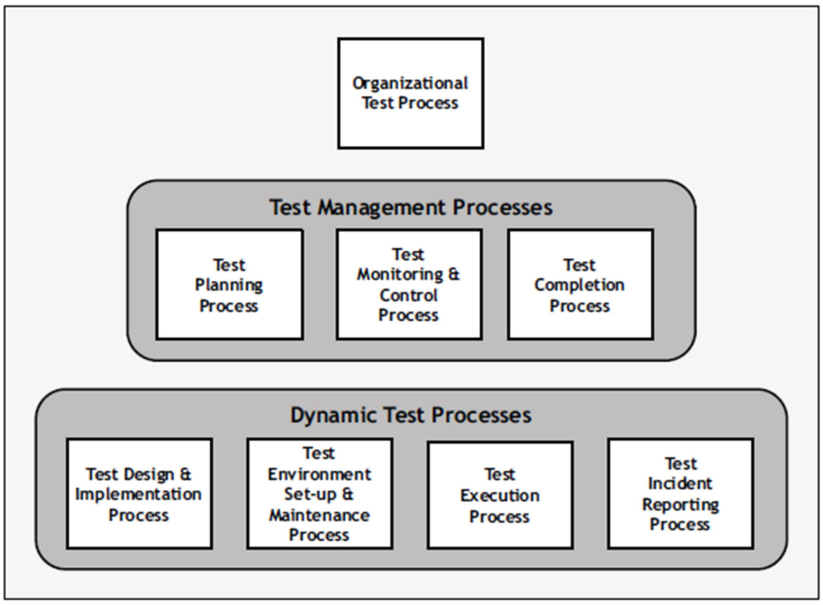
O processo de teste organizacional define um processo para a criação e manutenção de especificações de teste organizacional, como políticas de teste organizacional, estratégias, processos, procedimentos e outros ativos.

Os processos de gerenciamento de teste especificam os processos que cobrem o gerenciamento de testes para um projeto de teste completo, em qualquer fase de teste ou tipo de teste, dentro de um projeto de teste. Compreendem o processo de planejamento de teste, processo de monitoramento e controle de teste e o processo de conclusão do teste.

Os processos de teste dinâmico determinam os processos genéricos para realização de testes dinâmicos. São compostos pelos seguintes processos: Processo de teste e implementação de teste; Processo de configuração e manutenção do ambiente de teste; Processo de execução do teste; e processo de relatório de incidentes de teste.

As camadas do modelo de processo de teste compreendem números variáveis de processo de teste, como mostrado na Figura 12.

Figura 12: Modelo multicamadas que mostra todos os processos de teste



Fonte: ISO / IEC / IEEE 29119-2 (2013)

2.5.2.2 Processo de teste organizacional

O processo de teste organizacional é usado para desenvolver e gerenciar especificações de teste organizacionais, que normalmente se aplicam ao teste em toda a

organização. Inclui atividades para a criação, revisão e manutenção de especificações de teste organizacional, com o objetivo de desenvolver, monitorar a conformidade e manter especificações de teste organizacionais, tais como a Política de Teste Organizacional e Estratégia de Teste organizacional (ISO / IEC / IEEE 29119-2, 2013).

A Política de Teste Organizacional é um documento de nível executivo que descreve a finalidade e os objetivos dos testes dentro da organização. Também estabelece práticas de teste na organização e fornece um quadro para a elaboração, revisão e melhoria contínua da política de teste da organização, estratégia de teste e abordagem do gerenciamento de projetos de teste. Já a Estratégia de Teste Organizacional é um documento detalhado, técnico que define como o teste é realizado dentro da organização. É um documento genérico que fornece diretrizes para uma série de projetos na organização e não é específico do projeto.

A estratégia de teste organizacional precisa estar alinhada com a política de teste organizacional. O *feedback* desta atividade é direcionada para a política de teste para possível melhoria de processos. Da mesma maneira, os processos de gestão de teste a serem utilizados em cada um dos projetos dentro da organização também precisam se integrar com a estratégia e política organizacional de teste.

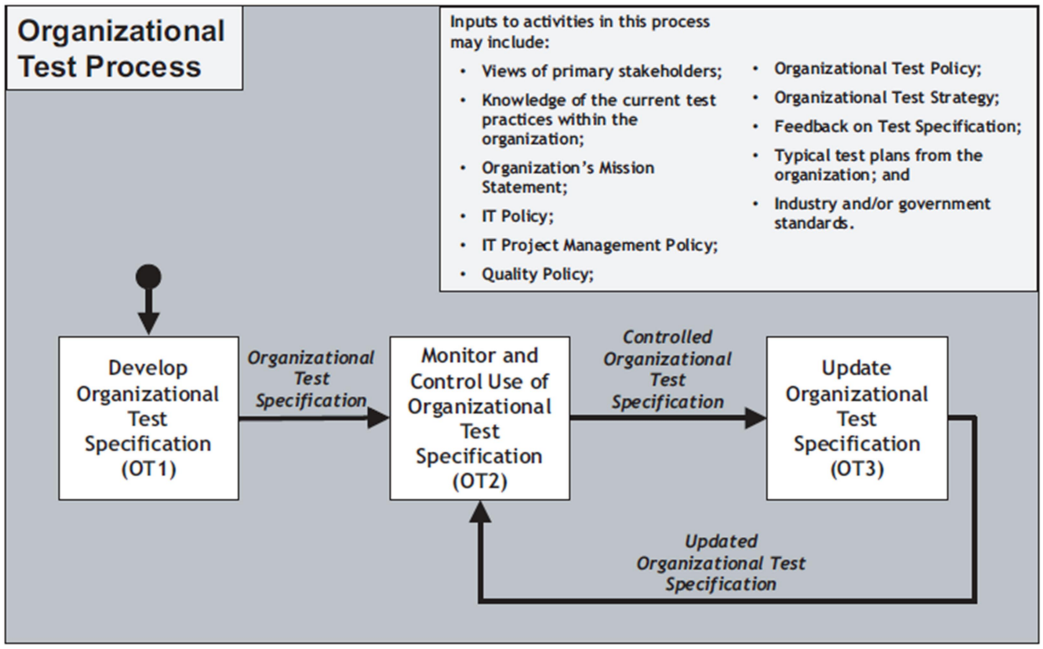
O responsável pelas especificações de teste organizacional deve implementar as seguintes atividades e tarefas de acordo com as políticas e procedimentos de organização aplicáveis com relação ao Processo de Teste Organizacional (ISO / IEC / IEEE 29119-2, 2013):

1. Desenvolver a especificação do teste organizacional (OT1): Os requisitos para as especificações de testes organizacionais devem ser identificados, por meio de análise de documentos de origem relevantes, a partir de oficinas, entrevistas ou outros meios adequados. A aprovação do conteúdo da especificação do teste organizacional deve ser obtida junto das partes interessadas.
2. Monitorar e Controlar o Uso da Especificação de Teste Organizacional (OT2): O uso da Especificação do Teste Organizacional deve ser monitorado para determinar se ele está sendo usado efetivamente dentro da organização. Devem ser tomadas medidas adequadas para encorajar o alinhamento das partes interessadas.
3. Atualizar a Especificação do Teste Organizacional (OT3): Os comentários sobre o uso da especificação do teste organizacional devem ser revisados. A eficácia do uso e gerenciamento da especificação do teste organizacional deve ser considerada e todos os comentários e mudanças para melhorar sua eficácia devem ser determinados e aprovados. Isso pode ser alcançado por meio de revisão de comentários, workshops, entrevistas e outros meios adequados. Onde as mudanças na especificação do teste

organizacional foram identificadas e aprovadas, essas alterações devem ser implementadas.

Como resultado do sucesso da implementação do processo de teste organizacional são identificados os requisitos para especificações de teste organizacional, as especificações de teste organizacional são desenvolvidas, a conformidade com as especificações de teste organizacional é monitorada, as atualizações para especificações de teste organizacionais são acordados pelas partes interessadas e realizadas, entre outras melhorias. A visão geral deste processo é ilustrada na Figura 13.

Figura 13: Processo de teste organizacional



Fonte: ISO / IEC / IEEE 29119-2 (2013)

2.5.2.3 Processos de gerenciamento de testes

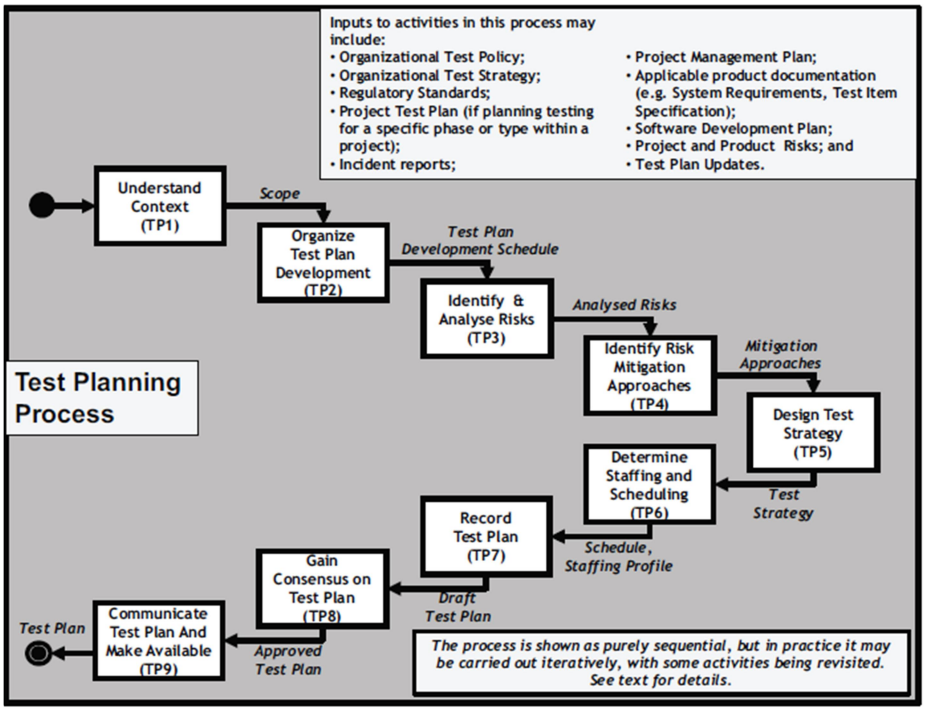
Os processos de gerenciamento de testes são compostos pelo planejamento de testes, monitoramento e controle de testes e conclusão do teste. Esses processos genéricos podem ser aplicados no nível do projeto, para gerenciamento de testes em diferentes fases de teste e para gerenciar vários tipos de teste. Quando aplicados no nível de gerenciamento de teste do projeto, são usados para gerenciar o teste para todo o projeto, com base em um plano de teste do projeto. Para muitos projetos, cada um dos testes exigirá que os processos de gerenciamento de teste sejam aplicados à sua gestão separadamente (ISO / IEC / IEEE 29119-2, 2013).

2.5.2.3.1 Processo de planejamento de teste

O objetivo do processo de planejamento de teste é desenvolver, concordar, registrar e comunicar as informações relevantes aos interessados, e abordagens que serão utilizadas no teste, permitindo a identificação precoce de recursos, ambientes e outros requisitos do teste.

No processo de planejamento de teste é desenvolvido o Plano de Teste. Para isso, as atividades mostradas na Figura 13 devem ser realizadas. Como o conteúdo do plano de teste se torna disponível a partir da realização das atividades definidas, um projeto de plano de teste será gradualmente elaborado até que o plano de teste completo seja gravado. Devido à natureza iterativa do processo, caso necessário as atividades mostradas na Figura 14 podem ser revistas antes que o plano de teste completo possa ser disponibilizado. Na maioria das vezes, as atividades TP3, TP4, TP5 e TP6 precisarão ser realizadas iterativamente para conseguir um plano de teste aceitável.

Figura 14: Processo de planejamento de teste



Fonte: ISO / IEC / IEEE 29119-2 (2013)

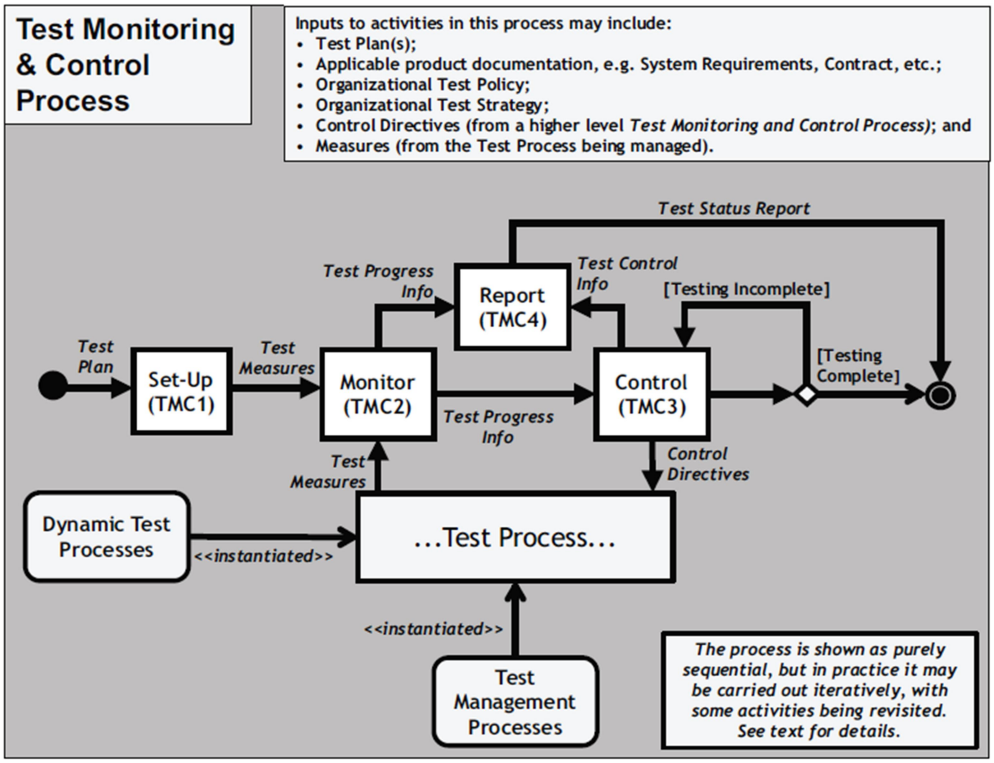
Durante o teste, o plano de teste poderá ser modificado em resposta aos resultados da implementação do plano e do surgimento de novas informações. Por exemplo, podem surgir novos riscos que ameaçam o projeto ou o produto, então o processo deve ser novamente inserido em Identificar e Analisar Riscos (TP3).

Como resultado da implementação bem-sucedida do Processo de Planejamento de Teste, o escopo do trabalho do projeto de teste é analisado e compreendido, as partes interessadas que participarão no planejamento do teste serão identificadas e informadas. Os riscos que podem ser tratados por testes são identificados, analisados e classificados com um nível acordado de exposição ao risco, identificam-se estratégias de teste, ambiente de teste, ferramenta de teste e necessidades de dados de teste. Cada atividade está programada, as estimativas são calculadas e as evidências para justificar as estimativas são registradas. O Plano de Teste é acordado e distribuído a todas as partes interessadas.

2.5.2.3.2 Processo de monitoramento e controle de testes

O objetivo do Processo de Monitoramento e Controle de Teste, como ilustrado na Figura 15, é determinar se o teste progride de acordo com o Plano de Teste e com as especificações de teste organizacional. Ele também inicia as ações de controle conforme necessário e identifica atualizações do Plano de Teste.

Figura 15: Processo de monitoramento e controle de testes



Fonte: ISO / IEC / IEEE 29119-2 (2013)

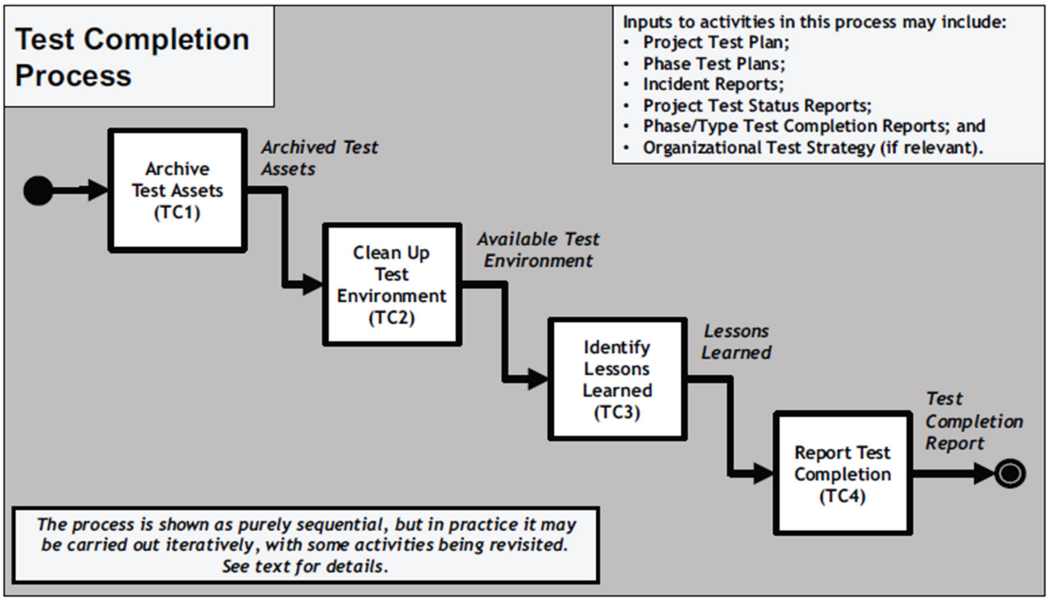
Como resultado da implementação bem sucedida do Processo de Monitoramento e Controle de Teste os meios de coleta de medidas adequadas para monitorar o progresso do teste e a mudança do risco são criados, o progresso em relação ao plano de teste é monitorado, novos riscos relacionados ao teste são identificados e analisados, as ações de

controle necessárias são identificadas e comunicadas às partes interessadas, a decisão de suspender o teste é aprovada, o progresso do teste e as mudanças nos riscos são relatados às partes interessadas. Por fim, são obtidos os relatórios de status de teste e as atualizações do plano de teste.

2.5.2.3.3 Processo de conclusão do teste

O processo de conclusão do teste, conforme mostrado na Figura 16, é realizado quando for acordado que as atividades de teste estão completas. Será realizado para completar os testes realizados em uma fase de teste específica ou tipo de teste e para completar o teste para um projeto completo. Seu objetivo é disponibilizar recursos de teste úteis para uso posterior, deixar o ambiente de teste em condições satisfatórias, registrar e comunicar os resultados do teste e informações relevantes aos *stakeholders*. Os recursos de teste incluem planos de teste, especificações do caso de teste, *scripts* de teste, ferramentas de teste, dados de teste e estrutura de ambiente de teste.

Figura 16: Processo de conclusão do teste



Fonte: ISO / IEC / IEEE 29119-2 (2013)

Como resultado da implementação bem-sucedida do processo de conclusão do teste, os recursos de teste são arquivados ou encaminhados diretamente para as partes interessadas, o ambiente de teste está em seu estado acordado, todos os requisitos de teste são satisfeitos e verificados, o relatório de conclusão do teste é aprovado e comunicado às partes interessadas.

2.5.2.3.4 Processos de teste dinâmico

Os Processos de Teste Dinâmico são usados para realizar ensaios dinâmicos dentro de uma fase de teste específica (unidade, integração, sistema e aceitação) ou tipo de teste (por exemplo, testes de desempenho, testes de segurança, teste de usabilidade). Existem quatro processos de teste dinâmicos, como mostra a Figura 15: processo de *design* e implementação de teste, processo de configuração e manutenção do ambiente de teste, processo de execução do teste e processo de relatório de incidentes de teste (ISO / IEC / IEEE 29119-2, 2013).

O processo de *design* e implementação de teste é usado para derivar casos de teste e procedimentos de teste, esses são normalmente documentados em uma especificação de teste, mas podem ser imediatamente executados quando, por exemplo, realiza-se o teste exploratório, caso em que eles não são suscetíveis de serem documentados com antecedência. Como resultado da realização deste processo, os seguintes itens de informações devem ser obtidos: especificações de teste (especificações de projeto de teste, especificações de caso de teste e especificações procedimento de teste) e informações de rastreabilidade relacionada; requisitos de dados de teste; os requisitos do ambiente de teste (ISO / IEC / IEEE 29119-2, 2013).

O processo de configuração e manutenção do ambiente de teste é usado para estabelecer e manter o ambiente em que os testes são executados. Manutenção do ambiente de teste pode envolver alterações com base nos resultados de testes anteriores, onde existem processos de mudança e gerenciamento de configuração, alterações no teste ambientes podem ser gerenciados usando esses processos. O objetivo é estabelecer e manter o necessário no ambiente de teste e para comunicar seu status para todas as partes interessadas. Como resultado da realização deste processo, os seguintes itens de informação devem ser produzidos: ambiente de teste; dados de teste e relatório de preparação do ambiente de teste (ISO / IEC / IEEE 29119-2, 2013).

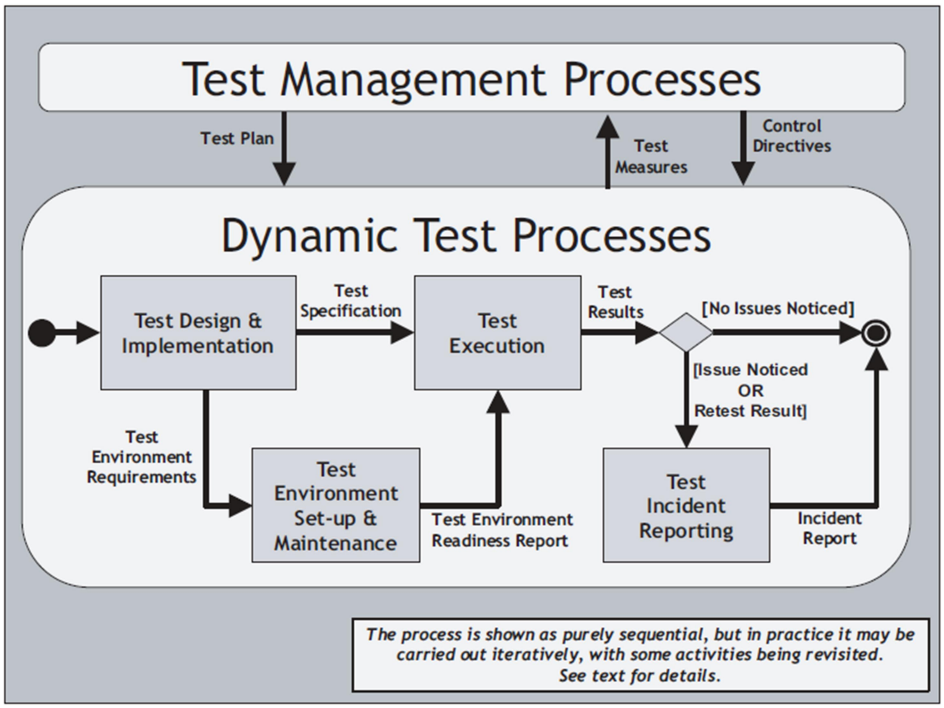
O processo de execução do teste é usado para executar os procedimentos de teste gerados como resultado do processo de *design* e do processo de configuração e manutenção do ambiente de teste. Como resultado do sucesso da implementação do processo de execução de teste os procedimentos dos testes são executados, os resultados reais são gravados, os resultados reais e esperados são comparados e os resultados do teste são determinados (ISO / IEC / IEEE 29119-2, 2013).

O processo de relatório de incidentes de teste é utilizado para a notificação de incidentes de teste. Este processo será inserido como resultado da identificação de falhas nos testes, casos em que algo incomum ou inesperado ocorreu durante execução do teste, ou quando

um novo teste passa. O objetivo é relatar os incidentes às partes interessadas e identificar necessidades de medidas adicionais como resultado da execução do teste. No caso de um novo teste isso exigirá um relatório de incidente a ser criado. No caso de um reteste, isso vai exigir o status de um incidente anteriormente levantada no relatório a ser atualizado, mas também pode exigir um novo relatório sobre o incidente a ser levantado (ISO / IEC / IEEE 29119-2, 2013).

Para qualquer teste específico, os processos de teste dinâmico serão executados na ordem mostrada na Figura 17, mas estes processos normalmente serão invocados várias vezes para completar o teste para uma determinada fase de teste (por exemplo, teste do sistema) ou tipo de teste (por exemplo, teste de desempenho). Isso ocorre porque, à medida que os testes são projetados e executados, o processo de gerenciamento de teste de supervisão monitora o progresso do teste e pode exigir testes adicionais para serem projetados e executados até o que critério de conclusão para esta atividade de teste seja alcançado.

Figura 17: Processos de teste dinâmicos



Fonte: ISO / IEC / IEEE 29119-2 (2013)

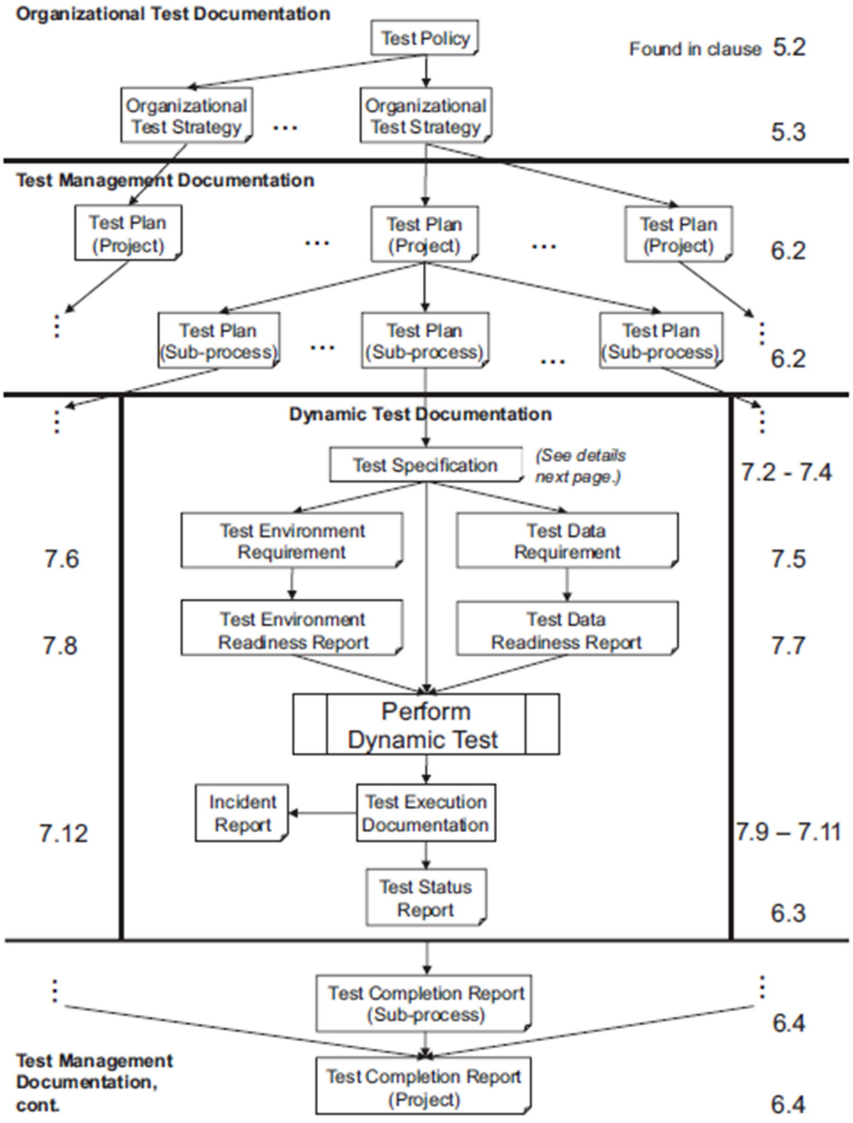
Medidas de teste, que são uma saída dos processos de teste dinâmico e uma entrada no monitoramento de testes e processo de controle, podem ser produzidas durante qualquer atividade dos processos de teste dinâmicos. As medidas de teste são usadas para relatar o status e o progresso dos testes para testar a equipe de gerenciamento. Da mesma maneira, as diretrizes de controle são uma saída do processo de gerenciamento de teste e uma

entrada para o teste dinâmico, e pode ser atuado durante qualquer atividade dos processos de teste dinâmico. As diretrizes correspondem a instruções da equipe de gerenciamento de testes que determinam como o teste dinâmico deve ser conduzido pela equipe de teste.

2.5.3 ISO / IEC / IEEE 29119-3: Documentação de teste

O objetivo da ISO / IEC / IEEE 29119-3 é definir modelos para documentação de teste que cobrem todo o ciclo de vida do teste de *software*. Todos os modelos se alinham com o processo de teste definido no ISO / IEC / IEEE 29119-2 e podem ser produzidos aplicando os processos definidos nesse padrão. Uma visão geral dos documentos é fornecida na Figura 18.

Figura 18: Visão geral dos documentos do processo de teste



Fonte: ISO / IEC / IEEE 29119-3 (2013)

Cada modelo pode ser adaptado para atender às necessidades exclusivas de cada organização, para suportar a implementação do padrão em qualquer modelo de ciclo de vida de desenvolvimento de *software* (ISO / IEC / IEEE 29119-3, 2013).

O padrão de documentação de teste IEEE 829, bem conhecido e amplamente utilizado, foi usado como base para este padrão e, como tal, o ISO / IEC / IEEE 29119-3 substitui o IEEE 829. Os documentos que são definidos no ISO / IEC / IEEE 29119-3 são referentes à documentação do processo de teste organizacional, documentação do processo de gerenciamento de teste documentação do processo dinâmico (ISO / IEC / IEEE 29119-3, 2013).

Os documentos descritos nesta parte da ISO / IEC / IEEE 29119 pode ser emitido em várias versões ao longo do tempo. No entanto, o tratamento de várias versões de documentos está fora do escopo desta parte da ISO / IEC / IEEE 29119, porque esta é uma questão de gestão de configuração.

2.5.3.1 Documentação do processo de teste organizacional

As especificações de teste organizacional descrevem informações sobre testes no nível da organização e não são dependentes do projeto. Exemplos típicos de especificações de teste organizacional desenvolvidas no processo de teste organizacional incluem a política de teste e estratégia de teste organizacional.

A política de teste define os objetivos e os princípios do teste de *software* a serem aplicados na organização. Ela especifica o que deve ser feito testando, mas não detalha como o teste é realizado. A política fornece uma estrutura para estabelecer, revisar e melhorar continuamente a política de teste da organização (ISO / IEC / IEEE 29119-3).

O conteúdo da Política de Teste inclui informações específicas do documento, que descreve suas origens, histórico, autores, aprovadores e identifica de modo exclusivo uma versão do documento. Fornece informações explicativas sobre o contexto e a estrutura do documento, identificando a extensão da cobertura da área de assunto pelo documento e descreve quaisquer inclusões, exclusões, premissas ou limitações. Apresenta as declarações de política de teste, descrevendo a finalidade, os objetivos, o escopo geral dos testes dentro da organização, identificando o processo de teste que a organização seguirá, além de conter treinamentos e o código de ética organizacional a ser confirmado pelos testadores, entre outras informações relevantes.

A Estratégia de Teste Organizacional é um documento técnico que fornece diretrizes sobre como o teste deve ser realizado dentro da organização, ou seja, como alcançar os objetivos estabelecidos na Política de Teste (ISO / IEC / IEEE 29119-3). Para organizações

pequenas ou altamente homogêneas, uma única Estratégia de Teste Organizacional pode abranger todos os testes. Uma organização pode ter mais de uma estratégia de teste organizacional se a organização executar o desenvolvimento de várias maneiras significativamente diferentes, como produtos críticos de segurança e produtos não críticos, ou se estiver usando modelos de desenvolvimento ágil e modelo V, ou se seus programas são grandes o suficiente para merecer sua própria estratégia.

O conteúdo da estratégia de teste organizacional, contém as declarações de estratégia de teste organizacional em todo o projeto, o gerenciamento de risco genérico, identificando a abordagem genérica do gerenciamento de riscos que se espera que seja usada para direcionar as atividades de teste. Descreve a abordagem da organização para selecionar e priorizar a execução do teste. Os procedimentos de teste consistem em casos de teste priorizados, derivados de conjuntos de recursos priorizados por meio de condições de teste priorizadas e itens de cobertura. A documentação e relatório de teste identificam os documentos que devem ser produzidos durante o teste para o projeto de teste como um todo. Detalha também, a abordagem para testar a automação dentro da organização, identificando as ferramentas de teste a serem usadas durante o teste, entre outras informações relevantes (ISO / IEC / IEEE 29119-3).

2.5.3.2 Documentação de processos de gerenciamento de teste

Os documentos desenvolvidos nos processos de gerenciamento de testes compreendem os seguintes tipos: plano de teste; relatório de status do teste; relatório de conclusão do teste.

O plano de teste fornece um documento de gerenciamento de testes. Alguns projetos podem ter um único plano de teste, enquanto que para projetos maiores, vários planos de teste podem ser produzidos. Ele descreve as decisões tomadas durante o planejamento inicial e evolui à medida que o replanejamento é realizado como parte da atividade de controle. O conteúdo do plano de teste inclui informações específicas do documento, identifica os projetos ou os subprocesso de teste para o qual o plano está sendo escrito e outras Informações contextuais relevantes. Também identifica quaisquer características dos itens de teste que devem ser excluídos especificamente do teste e a justificativa de sua exclusão. Identifica os riscos e fornece um nível de exposição para cada risco com base em seu impacto e probabilidade, apresentando recomendações para tratar os riscos (ISO / IEC / IEEE 29119-3).

O relatório de status do teste descreve informações sobre o status do teste que é realizado em um período de relatório específico. Em um projeto ágil, o relatório de status do

teste pode não ser um documento escrito. Por exemplo, seu conteúdo poderia ser discutido em reuniões de iteração e complementado por informações armazenadas em fóruns de atividades e gráficos. Apresenta o progresso que tem sido feito contra o plano de teste. Quaisquer desvios notáveis do plano devem ser destacados, com explicações sobre as razões para o desvio e descrição de quaisquer ações corretivas. Identifica os fatores que impediam o progresso durante o período do relatório e as soluções correspondentes que foram implementadas para removê-los. Lista os novos riscos que foram identificados como resultado do monitoramento e controle de teste, bem como alterações aos riscos existentes durante o período de reporte. Descreve ainda, o teste previsto para o próximo período de referência (ISO / IEC / IEEE 29119-3).

O relatório de conclusão do teste fornece um resumo do teste que foi realizado. Isto pode ser para o projeto, programa como um todo ou para a subprocesso de teste particular. O conteúdo do relatório de conclusão do teste descreve detalhes sobre o que foi testado e quaisquer limitações sobre a maneira como o teste foi realizado.

2.5.3.3 Documentação de processos de teste dinâmico

Os documentos desenvolvidos nos processos de teste dinâmico compreendem os seguintes tipos: Especificação de teste, divididos em: especificação do projeto de teste, especificação do caso de teste e especificação do procedimento de teste; Requisitos de dados de teste; Requisitos de ambiente de teste; Relatório de prontidão de dados de teste; Relatório de preparação do ambiente de teste; Documentação de Execução de Teste, dividida em: Resultados reais, Resultados do teste e Registro de Execução de Teste; Relatório de incidentes (ISO / IEC / IEEE 29119-3).

A especificação de projeto de teste identifica os recursos a serem testados e as condições de teste derivadas do teste base para cada uma das características como o primeiro passo para a definição de casos de teste e procedimentos de teste para serem executados. Um conjunto de recursos é um agrupamento lógico dos recursos a serem testados para os itens de teste, que são especificados no Plano de Teste.

A especificação do caso de teste evidencia os itens de cobertura de teste e os casos de teste correspondentes derivados da base de teste para um ou mais conjuntos de recursos. A cobertura do teste, refere-se ao que é esperado para ser coberto por um caso de teste de acordo com a técnica de *design* que foi usado durante a sua derivação. Um caso de teste especifica como um ou mais itens de cobertura de teste são exercidos para ajudar a determinar se a parte do item de teste foi ou não implementado corretamente. Especifica os

resultados esperados e os comportamentos necessários do item de teste em resposta às entradas que são dadas.

A especificação do procedimento de teste descreve os casos de teste nos conjuntos de teste selecionados na ordem de execução, juntamente com as ações associadas que podem ser necessárias para configurar as condições prévias iniciais e quaisquer atividades de encerramento de execução posterior.

Os requisitos de dados de teste apresentam as propriedades dos dados de teste necessários para executar os procedimentos de teste definido na Especificação do Procedimento de Teste. Os requisitos do ambiente de teste caracterizam as propriedades do ambiente de teste necessário para executar os procedimentos de teste definidos na Especificação do Procedimento de Teste.

O relatório de preparação de dados de teste descreve o cumprimento de cada requisito de dados de teste. Enquanto o relatório de preparação do ambiente de teste, apresenta o cumprimento de cada requisito de ambiente de teste.

Os resultados reais são um registro do resultado da execução de um caso de teste em um procedimento de teste. Os resultados reais são comparados com os resultados esperados para determinar o resultado do teste. Os resultados reais nem sempre são registradas formalmente, para alguns tipos de sistemas pode ser necessário para documentar plenamente os resultados reais, e em outros podem optar por fazerem a gravação integral dos resultados reais. A gravação pode ser feita por uma ferramenta automatizada durante a execução do teste.

O resultado do teste é um registro de se uma execução específica caso de teste foi aprovada ou rejeitada, ou seja, se os resultados reais correspondem aos resultados esperados ou se desvios foram observados, ou se a prevista execução de caso de teste não foi possível. O resultado do teste para um caso de teste geralmente é registrado diretamente no procedimento de teste em um espaço reservado para este propósito. O resultado do teste, portanto, não é geralmente considerado como um documento independente.

O registro de execução de teste (Log) grava detalhes da execução de um ou mais procedimentos de teste. Os procedimentos de teste podem ser descritos em listas ou em tabelas em um documento ou produzidos por uma ferramenta de teste e um banco de dados.

Um incidente de teste é qualquer problema que é notado durante o teste que exige que as ações sejam documentadas. Os incidentes de teste são registrados em relatórios de incidentes. Haverá um relatório de incidente para cada incidente único (Os relatórios também podem ser conhecidos como relatórios de defeitos, relatórios de erros, relatórios de falhas, etc.).

2.5.4 ISO / IEC / IEEE 29119-4: Técnicas de teste

O objetivo das Técnicas de Teste ISO / IEC / IEEE 29119-4 é definir um padrão internacional que abranja técnicas de teste de *software* que podem ser usadas durante o projeto de teste e processo de implementação, dentro de qualquer organização ou modelo de ciclo de vida de desenvolvimento de *software*, fornecendo aos *stakeholders* a capacidade de projetar casos de teste para o teste de *software* em qualquer organização (ISO / IEC / IEEE 29119-4, 2015).

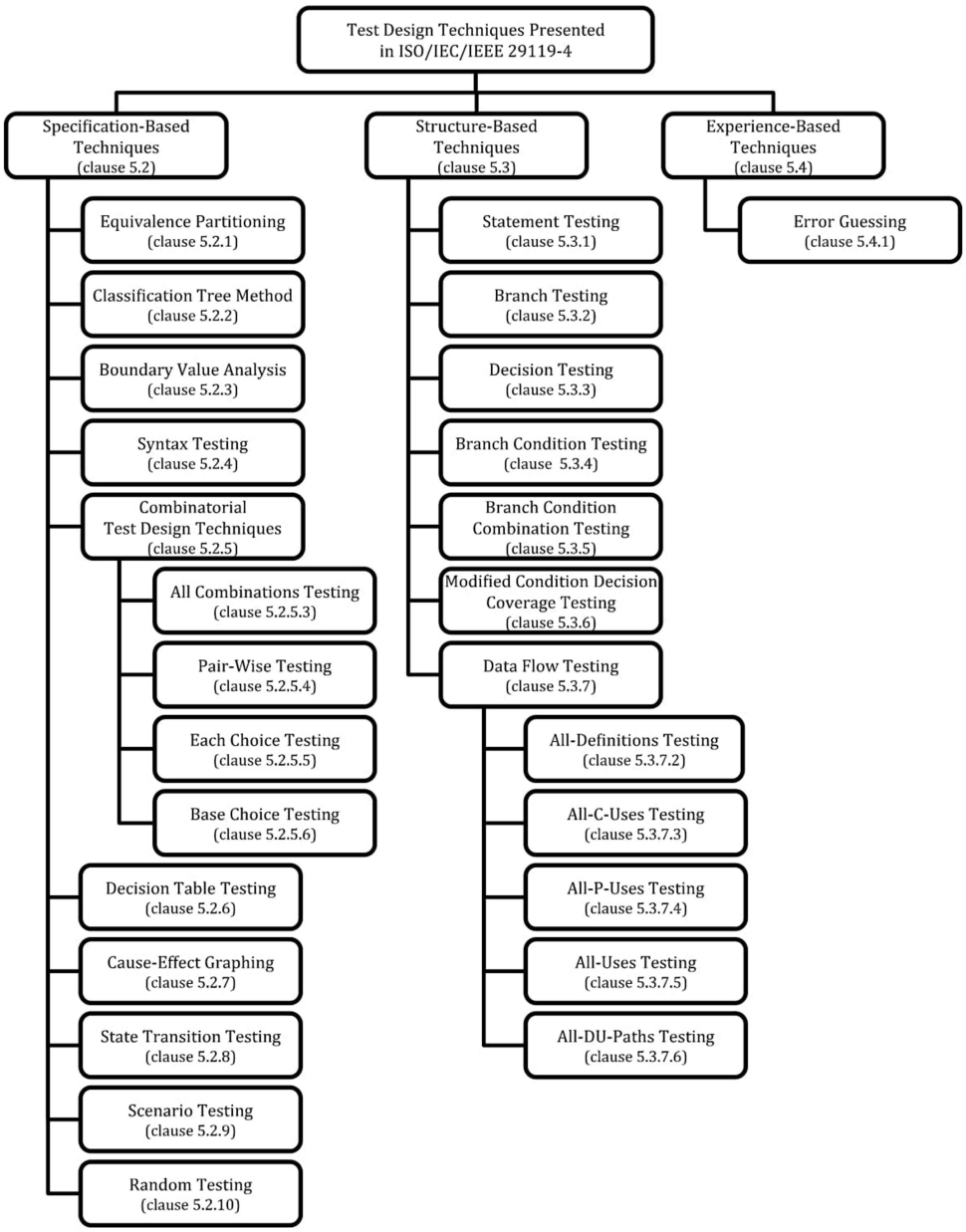
Esta parte da ISO / IEC / IEEE 29119 não prescreve um processo para a concepção e implementação de testes, e sim descreve um conjunto de técnicas que podem ser usadas dentro da ISO / IEC / IEEE 29119-2. A intenção é descrever uma série de técnicas que têm ampla aceitação na indústria de testes de *software*.

A ISO / IEC / IEEE 29119-4 define técnicas de projeto de teste para testes baseados em especificação, testes baseados em estrutura e testes baseados em experiência. Nos testes baseados na especificação, a base do teste (por exemplo, requisitos, especificações, modelos ou necessidades do usuário) é usada como a principal fonte de informação para a concepção de casos de teste. Em testes baseados em estrutura, a estrutura do item de teste (por exemplo, código fonte ou a estrutura de um modelo) é usada como fonte primária de informação para criar casos de teste. Nos testes baseados em experiência, o conhecimento e a experiência do testador são utilizados como fonte primária de informações durante o *design* do caso de teste. Para testes baseados em especificação, testes baseados em estrutura e testes baseados na experiência, a base do teste é usada para gerar os resultados esperados. Essas classes de técnicas de teste são complementares e sua aplicação combinada geralmente resulta em testes mais efetivos (ISO / IEC / IEEE 29119-4, 2015).

Os termos "testes de caixa preta" e "teste de caixa branca" utilizados nesse padrão, referem-se à visibilidade da estrutura interna do item de teste. No teste de caixa preta, a estrutura interna do item de teste não é visível, enquanto que para o teste de caixa branca, a estrutura interna do item de teste está visível. Quando uma técnica é aplicada ao utilizar uma combinação de conhecimento da especificação e estrutura dos itens de teste, denomina-se "teste de caixa cinza".

As técnicas que são descritas na ISO / IEC / IEEE 29119-4 são mostradas na Figura 19. Este conjunto de técnicas não é exaustivo. Das atividades no projeto de teste e implementação, as técnicas de projeto de teste fornecem orientação única e específica sobre a derivação das condições de teste, itens de cobertura de teste e casos de teste. Portanto, cada técnica é descartada em termos dessas três atividades.

Figura 19: O conjunto de técnicas de projeto de teste



Fonte: ISO / IEC / IEEE 29119-4 (2015)

No passo de criação de caso de teste de cada técnica, os casos de teste que são criados podem ser válidos, ou seja, eles contêm valores de entrada que o item de teste deve aceitar como correto, ou inválido, contêm pelo menos um valor de entrada que o teste rejeita e classifica como incorreto, idealmente com uma mensagem de erro apropriada.

2.5.4.1 Técnicas de projeto de teste baseadas em especificações

2.5.4.1.1 Partição de equivalência

A partição de equivalência usa um modelo do item de teste que divide as entradas e saídas do item de teste em partições de equivalência, também chamadas de "partições" ou "classes de equivalência", onde cada partição de equivalência deve ser declarada como uma condição de teste. Essas partições de equivalência devem ser derivadas da base de teste, onde cada partição é escolhida de modo que todos os valores dentro da partição de equivalência possam razoavelmente ser tratados de maneira semelhante pelo item de teste. As partições de equivalência podem ser derivadas para entradas e saídas válidas e inválidas (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.1.2 Método da árvore de classificação

O método da árvore de classificação usa um modelo do item de teste que divide as entradas do item de teste e os representa graficamente como uma árvore chamada árvore de classificação. As entradas dos itens de teste são divididas em classificações, onde cada classificação consiste em um conjunto de classes disjuntas e muitas vezes subclasses, todas as classificações de todas as entradas relevantes para o domínio do item de teste que está sendo modelado foram identificadas. Cada classificação deve ser uma condição de teste. As classes que resultam da decomposição das classificações podem ser divididas ainda mais em subclasses dependendo do nível de rigor exigido no teste. As relações hierárquicas entre classificações, classes e classes secundárias são modeladas como uma árvore, na qual o domínio de entrada do item de teste é colocado como o nó da raiz, as classificações como nós de ramificação e as classes ou subclasses como nós de folha (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.1.3 Análise do valor limite

A análise do valor de limite usa um modelo do item de teste que particiona as entradas e saídas do item de teste em uma série de conjuntos e subconjuntos (partições e subpartimentos) com limites identificáveis, onde cada limite é uma condição de teste. Os limites devem ser derivados da base do teste. Para os limites de saída, as partições de entrada correspondentes são derivadas com base no processamento descrito na especificação dos itens de teste. As entradas de teste são então selecionadas das partições de entrada (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.1.4 Teste de sintaxe

Teste de sintaxe usa um modelo formal das entradas para um item de teste como base para o projeto do teste. Este modelo de sintaxe é representado como uma série de regras, onde cada regra define o formato de um parâmetro de entrada em termos de "sequências de", "iterações de" ou "seleções entre" elementos na sintaxe. A sintaxe pode ser representada em um formato textual ou diagramático. A condição de teste no teste de sintaxe deve ser o modelo total ou parcial das entradas para o item de teste (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.1.5 Técnicas de projeto de teste combinatório

As técnicas de projeto de teste combinatório são usadas para derivar sistematicamente um subconjunto significativo e gerenciável de casos de teste que cobrem as condições de teste e os itens de cobertura de teste que são deriváveis durante o teste. As combinações de interesse são definidas em termos de parâmetros do item de teste e os valores que esses parâmetros podem tomar. Onde vários parâmetros devem interagir, esta técnica permite uma redução significativa no número de casos de teste necessários sem comprometer a cobertura funcional (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.1.6 Teste da tabela de decisão

Teste de tabela de decisão usa um modelo de relações lógicas (regras de decisão) entre as condições (causas) e as ações (efeitos) para o item de teste em uma tabela de decisão, onde cada condição booleana define um par de partições de equivalência de entrada para o item de teste, um correspondente ao caso "verdadeiro", e um ao caso "falso". Cada ação é um resultado esperado ou uma combinação de resultados para o item de teste expressado como um booleano. E um conjunto de regras de decisão define as relações necessárias entre as condições e as ações (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.1.7 Representação gráfica de causa

O grafo de efeito de causa usa um modelo de relações lógicas (regras de decisão) entre causas (por exemplo, entradas) e efeitos (saídas) para o item de teste, onde cada causa booleana define um par de partições de equivalência de entrada para o item de teste, um correspondente para o caso "verdadeiro" e um para o caso "falso"; e cada efeito define uma condição de saída esperada ou combinação de condições de saída para o teste, expresso como booleano (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.1.8 Teste de transição do estado

O teste de transição do estado usa um modelo dos estados que o item de teste pode ocupar as transições entre os estados, os eventos que causam transições e as ações que podem resultar das transições. Os estados do modelo devem ser discretos, identificáveis e finitos em número. Uma transição individual pode ser limitada por um guarda de eventos, que define um conjunto de condições que devem ser verdadeiras quando o evento ocorre, para que a transição ocorra. Nos testes de transição do estado, as condições do teste podem ser todos os estados do modelo de estado, todas as transições do modelo de estado ou todo o modelo de estado, dependendo dos requisitos de cobertura dos testes. O modelo pode ser representado como um diagrama de transição de estado ou uma tabela de estado (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.1.9 Testes de cenário

O teste de cenários usa um modelo das sequências de interações entre o item de teste e outros sistemas, com o objetivo de testar os fluxos de uso envolvendo o item de teste. As condições de teste devem ser uma sequência de interações, ou seja, um cenário, ou todas as sequências de interações. No teste de cenário, esta etapa deve incluir a identificação do cenário principal, que é a sequência típica de ações que se espera do item de teste ou uma escolha arbitrária quando nenhuma sequência típica de ações é conhecida (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.1.10 Teste aleatório

Teste aleatório usa um modelo do domínio de entrada do item de teste que define o conjunto de todos os valores de entrada possíveis. Deve ser escolhida uma distribuição de entrada para a geração de valores de entrada aleatórios. Todo o domínio de entrada deve ser a condição de teste para testes aleatórios (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.2 Técnicas de projeto de teste baseadas em estrutura

2.5.4.2.1 Teste de declaração

Um modelo do código fonte do item de teste que identifica declarações como executáveis ou não executáveis deve ser derivado. Cada declaração executável deve ser uma condição de teste. Cada declaração executável deve ser um item de cobertura de teste, ou seja, os itens de cobertura de teste são os mesmos que as condições de teste. Portanto,

nenhuma ação adicional é necessária nesta etapa para esta técnica (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.2.2 Teste de ramo

Um modelo de fluxo de controle que identifica os ramos *(branchs)* no fluxo de controle do item de teste deve ser derivado. Cada ramo no modelo de fluxo de controle deve ser uma condição de teste. Um ramo é uma transferência condicional de controle de qualquer nó no modelo de fluxo de controle para qualquer outro nó. O teste de ramo completo que cobre 100% de todos os ramos requer todos os arcos (links ou bordas) no gráfico de fluxo de controle a ser testado, incluindo instruções sequenciais entre um ponto de entrada e saída que não contém decisões. O teste de filial pode exigir testes de ramos condicionais e incondicionais, incluindo pontos de entrada e saída para um item de teste, dependendo do nível de cobertura de teste exigido (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.2.3 Teste de decisão

Um modelo de fluxo de controle do item de teste que identifica as decisões deve ser derivado. As decisões são pontos no item de teste em que dois ou mais resultados possíveis que podem ser tomados pelo fluxo de controle. As decisões típicas são usadas para seleções simples, para decidir quando sair de loops. No teste de decisão, cada decisão no modelo de fluxo de controle deve ser uma condição de teste (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.2.4 Teste de condição de ramo

Um modelo de fluxo de controle do item de teste que identifica decisões e condições deve ser derivado. No teste de condição de ramificação, cada decisão deve ser uma condição de teste. Deve-se identificar os caminhos de fluxo de controle que cobrem um ou mais itens de cobertura de teste que ainda não foram executados durante o teste e determinar as entradas de teste que fará com que os caminhos de fluxo de controle sejam exercidos. Seguem as mesmas ideia de derivação o teste de combinação de condição de ramo e o teste de cobertura de decisão de condição modificada (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.2.5 Teste de fluxo de dados

No teste de fluxo de dados, deve derivar-se um modelo do item de teste que identifique subdocumentos de fluxo de controle a partir do item de teste, dentro do qual cada definição de uma determinada variável está ligada a uso subsequente da mesma variável e dentro de que não houve uma redefinição interativa do valor da variável. Um "uso" é uma ocorrência de uma variável que não possui um novo valor. Os "usos" podem ser mais distinguidos como "*p-uses*" (predicado-uso) ou "*c-uses*" (computação-uso). Uma *p-use* denota o uso de uma variável na determinação do resultado de uma condição (predicado) dentro de uma decisão, como um loop while, se então, outro, etc. Um uso c ocorre quando uma variável é usada como uma entrada para a computação da definição de qualquer variável ou de uma saída (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.3 Técnicas de projeto de teste baseadas na experiência

2.5.4.3.1 Erro de adivinhação

A adivinhação de erros envolve o projeto de uma lista de verificação de tipos de defeito que podem existir no item de teste, permitindo que o testador identifique entradas para o item de teste que podem causar falhas, se esses defeitos existem no teste item. Cada tipo de defeito deve ser uma condição de teste. A lista de verificação dos tipos de defeito pode ser derivada por vários meios, tais como taxonomias de erros conhecidos, informações contidas nos sistemas de gerenciamento de incidentes, de um conhecimento, experiência, compreensão dos testadores sobre os itens de teste ou itens de teste semelhantes (ISO / IEC / IEEE 29119-4, 2015).

2.5.4.4 Medida de cobertura de teste

As medidas de cobertura definidas nesta parte da ISO / IEC / IEEE 29119 são baseadas em diferentes graus de cobertura que podem ser alcançados por técnicas de projeto de teste. Os níveis de cobertura podem variar de 0% a 100%. Em cada cálculo de cobertura, uma série de itens de cobertura de teste podem ser inviáveis. Um item de cobertura de teste deve ser definido como inviável se puder ser mostrado como não executável ou impossível de ser coberto por um caso de teste (ISO / IEC / IEEE 29119-4, 2015).

Esta parte também fornece definições informativas de uma variedade de tipos de testes relacionados à qualidade, contidas nos anexos e disponíveis para consulta.

2.5.5 ISO / IEC / IEEE 29119-5: Teste com palavras chave

O objetivo da norma ISO / IEC / IEEE 29119-5 é definir um padrão internacional para apoiar o teste de palavras chave, que se refere a uma maneira de descrever casos de teste usando um conjunto predefinido de palavras-chave. Uma ou mais palavras são usadas como uma referência a um conjunto específico de ações destinadas a serem realizadas durante a execução de um ou mais casos de teste. As ações incluem interações com a interface do usuário durante o teste, verificação e ações específicas para configurar um cenário de teste (ISO / IEC / IEEE 29119-5, 2016).

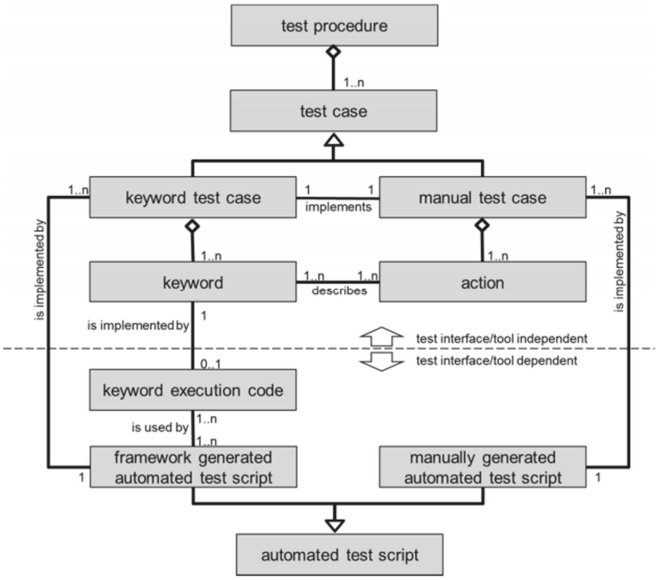
Em princípio, o teste de palavras chave pode ser aplicado a todos os níveis e tipos de testes. Incluem vários benefícios como, a facilidade de usar, compreensibilidade, manutenção, reutilização de informações de teste, suporte de automação de testes, potencial de economia de custo e cronograma.

A ideia fundamental do teste de palavras chave é fornecer um conjunto de blocos de construção, referido como palavras chave, que podem ser usados para criar casos de teste manuais ou automatizados sem a necessidade de conhecimento detalhado da programação ou experiência em ferramenta de teste. O objetivo final é fornecer um conjunto básico de palavras chave, abrangente o suficiente para que a maioria, se não todos, os casos de teste necessários possam ser inteiramente compostos por estas palavras chave.

Quando palavras chave não são utilizadas, casos de teste são geralmente escritos usando linguagem natural ou escritos em um computador com linguagem de programação. Comparado com a linguagem natural, palavras chave têm a vantagem de ser menos ambíguo e mais preciso. Comparado com uma linguagem de programação de computadores, quando palavras chave são bem definidas e estruturadas, elas têm a vantagem de ser compreensível por muitas pessoas que não têm habilidades de Engenharia de *Software* (ISO / IEC / IEEE 29119-5, 2016).

Um procedimento de teste pode ter múltiplos casos de teste nele, e um caso de teste pode, por sua vez, fazer parte de diferentes procedimentos de teste, como ilustrado na Figura 20. Um caso de teste é composto por ações de teste, palavras chave representam ações de teste. É possível mapear várias palavras chave para uma única ação, assim como é possível definir palavras chave de modo que cada palavra chave represente uma ação. A automação de teste é uma opção que pode ser escolhida ao implementar testes por palavra chave, mas também é possível utilizar uma abordagem manual. As palavras chave podem representar ações em diferentes níveis de abstração. Por exemplo, uma palavra chave pode se referir a um conjunto complexo de atividades, enquanto outra palavra chave pode se referir a uma ação muito simples (ISO / IEC / IEEE 29119-5, 2016).

Figura 20: Relações entre as entidades de teste dirigidas por palavras chave



Fonte: ISO / IEC / IEEE 29119-5 (2016)

O teste conduzido por palavras chave pode ser organizado usando uma ou mais camadas. As camadas típicas são o usuário final camada de domínio e a camada de interface de teste. As palavras chave na camada de domínio correspondem a atividades relacionadas a negócios ou domínio e refletem a terminologia usada por especialistas em domínio, geralmente são independentes da implementação. Palavras chave na camada de interface de teste referem-se a um tipo específico de interface de teste. As ações necessárias para abordar os itens de teste geralmente podem ser facilmente identificadas. O número total de palavras chave geralmente é menor do que na camada de domínio, uma vez que a interface de teste é limitada (ISO / IEC / IEEE 29119-5, 2016).

Para combinar as vantagens de várias camadas, por exemplo, camada de domínio e camada de interface de teste, uma estrutura é necessária, que pode ajudar a gerenciar palavras chave hierárquicas.

2.5.5.1 Tipos de palavras chave

As palavras chave simples, que são frequentemente usadas na camada de interface de teste, podem ser a conexão entre a ferramenta de execução de teste e palavras chave de nível superior em uma camada ou domínio intermediário. Usar apenas palavras chave na camada de interface de teste pode ser suficiente para a definição de casos de teste e sua execução, levando a testar casos com muitas ações.

As palavras chave simples são suficientes para compor e executar casos de teste, mas muitas vezes são insuficientes para refletir funcionalidades. Palavras chave compostas

são palavras chave compostas por outras palavras chave. Isso significa que as palavras chave podem ser organizadas em camadas diferentes.

As palavras chave podem ser classificadas em pelo menos duas categorias: etapas de navegação, isto é, entrada para o item de teste, e etapas de verificação ,ou seja, saída do item de teste. A maioria das palavras chave pertence à primeira categoria, pois muitas das ações são necessárias para preparar o item de teste ou realizar determinadas ações sobre ele, o que levará a um resultado. As etapas de navegação geralmente são passos que não verificam e registram o resultado do teste. Já as etapas de verificação estão relacionadas ao resultado do caso de teste (ISO / IEC / IEEE 29119-5, 2016).

As palavras chave podem ser usadas para determinar o estado do teste e para capturar os resultados do teste. Isso pode incluir a saída de teste, conformidade com os critérios de sucesso, testar arquivos de log de execução, saídas de *hardware*, status do sistema e falhas de teste (ISO / IEC / IEEE 29119-5, 2016).

O teste conduzido por palavras chave pode ser aprimorado se as palavras chave estiverem associadas aos dados. Para permitir uma associação com dados, em muitos casos, as palavras chave precisarão de parâmetros que possam ser corrigidos ou listados em lista. A maioria das palavras chave precisará ter pelo menos um parâmetro para especificar o objeto que eles aplicam.

2.5.5.2 Aplicação do teste dirigido por palavras chave

Ao identificar palavras chave, deve-se determinar as camadas necessárias no contexto dado e definir o tipo de palavras chave, além de identificar palavras chave na camada com base na definição ou escopo de cada camada. Geralmente, as palavras chave são definidas pela primeira identificação de conjuntos de ações que deverão ocorrer com frequência no teste. Uma palavra chave é descrita pelas seguintes informações: O nome da palavra chave, ele diz ao leitor o que essa palavra chave deve fazer; Os parâmetros da palavra chave, que podem estar vazios; E a documentação sobre a palavra chave, incluindo a camada em que esta palavra chave deve ser usada, a tipo de palavra chave, o contexto em que deve ser usado e as ações incluídas.

Ao se criar uma palavra chave, as seguintes questões devem ser consideradas (ISO / IEC / IEEE 29119-5, 2016):

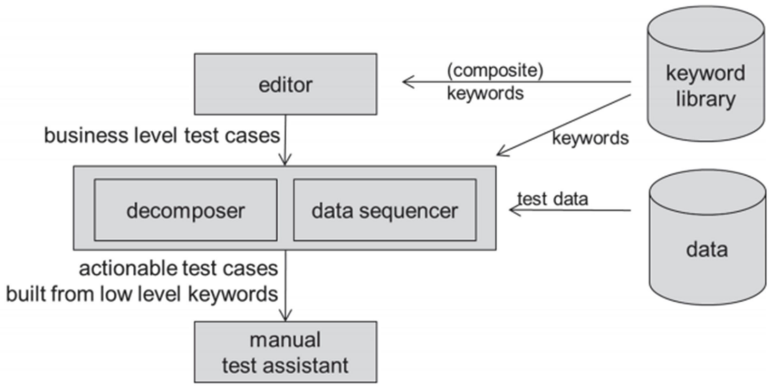
- Unicidade: cada palavra chave deve ser única em seu contexto de uso.
- Reutilização: as palavras chave devem ser definidas de modo a melhor suportar a reutilização futura.

- Completude: as palavras chave devem ser definidas com vista a todos os elementos conhecidos e possíveis interações da interface de teste (por exemplo, todos os objetos conhecidos na GUI e seus diálogos).
- Claridade: todas as palavras chave devem ser definidas com uma estrutura clara e consistente.
- Especificidade: as palavras chave não devem ser redundantes e devem ser mutuamente exclusivas para facilitar o desenho do teste e diminuir o esforço de manutenção.

Os casos de teste de palavras chave podem ser compostos por palavras chave previamente definidas. No processo de escrever casos de teste, pode ocorrer que as palavras chave faltantes sejam descobertas e, portanto, podem ser definidas após esse ponto.

Uma estrutura de teste orientada por palavras chave incluirá unidades funcionais que são mostradas em na Figura 21. O padrão não descreve como essas unidades funcionais devem ser implementadas. Uma ou mais ferramentas de *software*, juntamente com implementações personalizadas, bibliotecas e processos organizacionais, podem formar uma estrutura de teste dirigida por palavras chave.

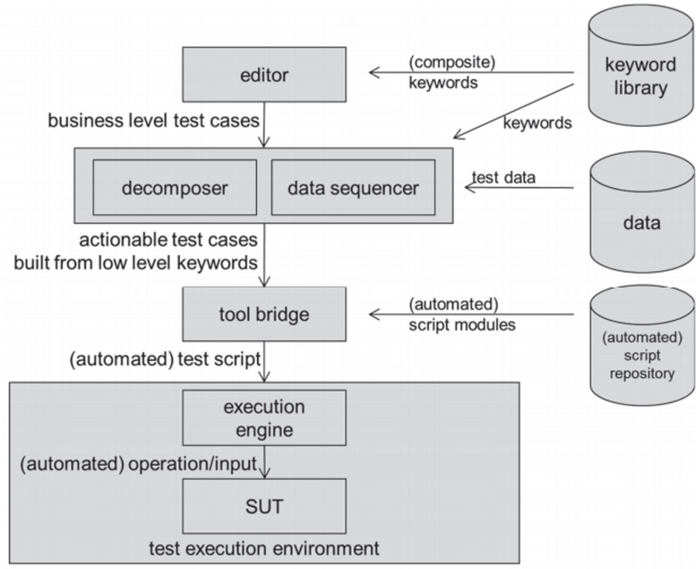
Figura 21: Componentes de uma estrutura de teste orientado por palavras chave para execução manual de teste



Fonte: ISO / IEC / IEEE 29119-5 (2016)

Na Figura 21, é mostrado um quadro de teste com palavra chave que é restrito ao suporte de testes manuais. Caso a automação de teste seja necessária, a estrutura seria composta de elementos adicionais, como mostrados na Figura 22. O assistente de teste manual não é necessário. Em vez disso, um mecanismo de execução é usado para executar os casos de teste. Uma ponte de ferramentas é usada como um link entre as palavras chave e sua representação no ambiente de execução de teste automatizado.

Figura 22: Componentes de uma estrutura de teste orientada por palavras chave para a execução automatizada de testes



Fonte: ISO / IEC / IEEE 29119-5 (2016)

A biblioteca de palavras chave armazena definições de palavras chave para um ou mais projetos ou partes desses projetos. É usado para armazenar as informações principais sobre palavras chave, tais como: nome, descrição, parâmetros e, no caso de palavras chave compostas, a lista de palavras chave a partir da qual a respectiva palavra chave é composta ou derivada.

O processo de teste definido na ISO / IEC / IEEE 29119-2 é aplicável a este padrão. Há várias razões para decidir converter casos de teste existentes em casos de teste dirigidos por palavras chave, entre elas: Garantia que todos os casos de teste tenham uma estrutura e estilos semelhantes irão melhorar a legibilidade; A manutenção futura pode ser mais barata se apenas um estilo de caso de teste for para ser mantido; Eficiência, as palavras chave identificadas nos casos de teste existentes podem ser reutilizáveis em casos de teste futuros; Automação, a mesma estrutura de automação pode ser usada para casos de teste antigos e novos; E por fim, a facilidade de compreensão, uma vez que os casos de teste podem ser usados e mantidos por testadores não necessariamente técnicos (ISO / IEC / IEEE 29119-5, 2016).

É importante ressaltar que este padrão possui vários anexos que auxiliam na implementação do teste orientado por palavras chave, como convenções de nomeação para palavras chave, benefícios que podem ser alcançados com os testes e conselhos sobre como as partes interessadas que desejam usar este tipo de teste podem iniciar as atividades.

3 DISCUSSÃO SOBRE AS ABORDAGENS DE MELHORIA DE PROCESSO DE TESTE DE SOFTWARE

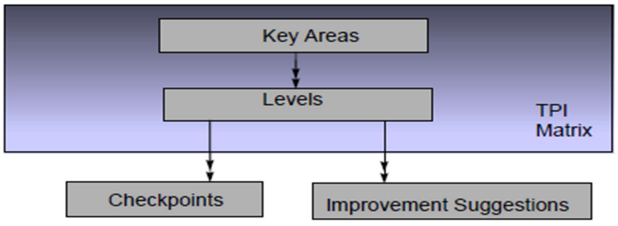
Abordagens de melhoria do processo de teste de *software* (STPI) são estruturas que orientam o desenvolvimento de *software* para melhorar o seu processo de teste de *software*. Porém, as expectativas das empresas diferem umas das outras dependendo, por exemplo, de metas internas, consciência da maturidade e conhecimento do processo. Assim, para melhorar o processo de teste de *software* de uma determinada organização, tem de ser encontrada uma abordagem adequada que se adapte às suas necessidades específicas e as metodologias (AFZAL et al., 2016).

Para desenvolver o conhecimento sobre as diferentes abordagens de melhoria do processo de teste de *software* existentes, bem como suas aplicações, foi realizado uma pesquisa de artigos devidamente publicados, a partir do ano de 2010, utilizando a ferramenta Google Acadêmico, com a seguinte string "*test process improvement*". Foram selecionados vinte artigos para estudos, os quais foram explorados e estratificados para uma planilha de controle, contendo os dados dos artigos, resumos, modelos de processo de *software*, entre outras informações que fossem avaliadas como interessantes. Diante dos resultados, foi possível observar que existem várias abordagens STPI, algumas com certas aplicações específicas, mas o que realmente as diferem são as representações do modelo. O estudo foi então aprofundado em duas abordagens típicas, muito mencionadas nos artigos em questão, o TPI e o TMMi.

3.1 Modelo TPI

O modelo TPI (*Test Process Improvement*), proposto por Tim Koomen e Martin Pol em 1998, oferece um instrumento para melhorar o processo de teste. O modelo oferece um quadro para determinar os pontos fortes e fracos do processo de teste atual dentro de um projeto ou organização, e oferece apoio na formulação de sugestões de melhoria adequadas para o processo de teste, em termos de qualidade, custo e tempo (TPI AUTOMOTIVE, 2004). O modelo TPI pode ser visualizado como apresentado na Figura 23.

Figura 23: Visão geral do modelo TPI



Fonte: TPI AUTOMOTIVE (2004)

O modelo TPI organiza o conhecimento em áreas chave. Uma área chave identifica um aspecto típico do processo de teste e oferece uma orientação para melhorar o processo de teste. Para cada área chave níveis de maturidade diferentes são encontrados, indicando maior maturidade. Cada nível seguinte difere em sua natureza de maturidade do anterior. Dentro de cada nível existem pontos de verificação e sugestões de melhorias.

A distribuição dos níveis de maturidade é baseada em prioridades e dependências entre os níveis de diferentes áreas chaves. Por exemplo, para começar com a automação de teste, é necessário ter casos de teste. Esses casos de teste podem ser derivados usando técnicas de *design* de teste.

Antes das sugestões de melhoria ser formulado, é necessário determinar o nível de maturidade de cada única área chave. Uma análise do processo de teste é realizada para recolher todas as informações necessárias para determinar se os pontos de verificação da áreas chave são cumpridos. Se sim, o nível de maturidade se aplica à área chave. A implementação das atividades de melhoria deve trazer a organização aos níveis de maturidade desejados para as respectivas áreas-chave.

O modelo TPI é organizado considerando 21 áreas chave que cobrem todos os aspectos de um processo de teste estruturado, conforme ilustrado no Quadro 3.

Quadro 3: Áreas chave do modelo TPI

Key Area	**Key Area**
Test strategy	Test functions and training
Life-cycle model	Scope of methodology
Moment of involvement	Communication
Estimation and planning	Reporting
Test design techniques	Defect management
Static test techniques	Testware management
Metrics	Test process management
Test automation	Evaluation
Test environment	Low-level testing
Office and laboratory environment	Integration testing
Commitment and motivation	

Fonte: TPI AUTOMOTIVE (2004)

As áreas chave são estruturadas em quatro grupos, que correspondem a diversas dimensões com as quais é necessário se preocupar durante o planejamento e a execução dos testes: infraestrutura, organização, ciclo de vida e tecnologia (TPI AUTOMOTIVE, 2004).

3.2 Modelo TMMi

Uma orientação que tem sido amplamente utilizada para melhorar a processos de desenvolvimento é o *Capability Maturity Model (CMM)*. O CMM e seu sucessor o *Capability*

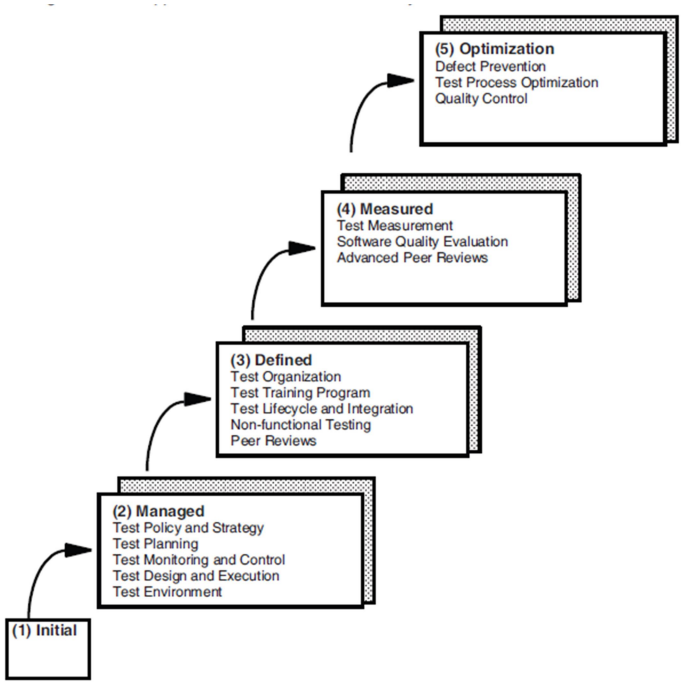
Maturity Model Integration (CMMI), são frequentemente considerados como o padrão da indústria para a melhoria do processo de *software*. Como o custo dos modelos de melhoria de processo de *software*, tais como o CMM e CMMI, são altos, a comunidade de testes criou seus próprios modelos de melhoria (TMMi FOUNDATION, 2010).

O TMMi (*Test Maturity Model integration*), é um modelo de maturidade de teste detalhado desenvolvido pela Fundação TMMi, para alcançar a melhoria do processo de teste, e está posicionada como sendo complementar ao CMMI. Assim como o CMMI, o TMMi também usa o conceito de níveis de maturidade para a avaliação do processo e melhoria (TMMi FOUNDATION, 2010).

Dentro do TMMi, o teste evolui de um processo caótico, mal definido e com a falta de recursos, para um processo maduro e controlado que tem a prevenção de defeitos como objetivo principal. O teste torna-se uma parte totalmente integrada do processo de desenvolvimento.

O TMMi tem sido desenvolvido como um modelo encenado para melhoria de processos, que usa conjuntos predefinidos de áreas de processo para definir um caminho de melhoria para uma organização. Este processo de melhoria é descrito por um componente chamado de nível de maturidade. Um nível de maturidade é uma evolução bem definida no sentido de alcançar melhoria organizacional processos.

Figura 24: Níveis de maturidade do TMMi e áreas de processo



Fonte: TMMi FOUNDATION (2010)

Há cinco níveis no TMMi que prescrevem uma hierarquia de maturidade e um caminho evolutivo para melhoria de processos de teste, como ilustrado na Figura 24. Cada nível tem um conjunto de áreas de processo que uma organização precisa implementar para alcançar a maturidade nesse nível, formando também uma base necessária para o próximo nível. A estrutura interna do TMMi é rica em práticas de teste que podem ser aprendidas e aplicadas de modo sistemático para apoiar um processo de testes de qualidade que melhora em passos incrementais (TMMi FOUNDATION, 2010).

No TMMi nível 1, a organização geralmente não fornece um ambiente estável para apoiar os processos. O sucesso nestas organizações depende da competência e heroísmo das pessoas na organização e não o uso comprovado de processos. O objetivo do teste aqui é mostrar que o *software* seja executado sem grandes falhas (TMMi FOUNDATION, 2010).

No TMMi nível 2, o teste torna-se um processo gerenciado e é claramente separado da depuração. No contexto da melhoria do processo de teste, uma estratégia de teste ou todo o programa de toda a empresa está estabelecido (TMMi FOUNDATION, 2010). Planos de teste são também desenvolvidos, técnicas de gestão de risco são usadas para identificar os riscos dos produtos com base em requisitos documentados.

No TMMi nível 3, o teste não é mais limitado a uma fase que segue após a codificação. É totalmente integrado no ciclo de vida de desenvolvimento. O planejamento de teste é feito numa fase inicial do projeto, por exemplo, durante a fase de requisitos, e está documentado em um plano de teste mestre. O desenvolvimento de um plano de teste mestre baseia-se em habilidades de planejamento de teste e os compromissos adquiridos no nível TMMi 2 (TMMi FOUNDATION, 2010).

Ao atingir as metas de nível TMMi 2 e 3, o teste pode se tornar um processo de medidas para incentivar o crescimento e realização. Em TMMi nível 4, o teste é um processo bem definido, bem fundamentado e mensurável. O teste é percebido como uma avaliação, que abrange todas as atividades do ciclo de vida relacionadas ao produto. Um programa de medição de teste para toda a organização é colocado em prática, para avaliar a qualidade do processo de teste, da produtividade e para monitorar melhorias (TMMi FOUNDATION, 2010).

A realização de todas as metas de melhoria de teste anteriores nos níveis 1 a 4 do TMMi cria uma infra-estrutura organizacional para o teste que suporta um processo completamente definido e medido. No nível TMMi nível 5, uma organização é capaz de melhorar continuamente seus processos com base em uma compreensão quantitativa dos processos controlados estatisticamente (TMMi FOUNDATION, 2010).

4 METODOLOGIA

Esta seção aborda os métodos definidos para o desenvolvimento do trabalho. Inicialmente foi realizado um estudo sobre Teste de *Software*, Métodos Ágeis, padrões da série ISO / IEC / IEEE 29119 e o levantamento de abordagens para a melhoria do processo de teste, como visto anteriormente.

Posteriormente, identificou-se a necessidade de uma aplicação prática referente ao entendimento da integração dos padrões com os métodos ágeis. Então, foi proposta a criação de um estudo de caso no laboratório iMobilis situado no Instituto de Ciências Exatas e Aplicadas (ICEA) da Universidade Federal de Ouro Preto. A ideia do estudo de caso é avaliar todo o processo de teste utilizado atualmente, documentá-lo e propor melhorias. O projeto foi aprovado pelo atual orientador de projetos do laboratório, em uma reunião ocorrida no dia 17/02/2017, e será divulgado nesta seção.

4.1 Estudo de Caso

A pesquisa de estudo de caso é uma técnica em que fatores chave são identificados e podem ter algum efeito no resultado e, em seguida, a atividade é documentada (WOHLIN *et al.,* 2012). É um método de observação, ou seja, é feito por observação de um projeto ou atividade. A ideia do estudo de caso do presente trabalho, é avaliar todo o processo de teste utilizado atualmente no laboratório iMobilis, documentá-lo, e propor melhorias.

4.1.1 Laboratório iMobilis

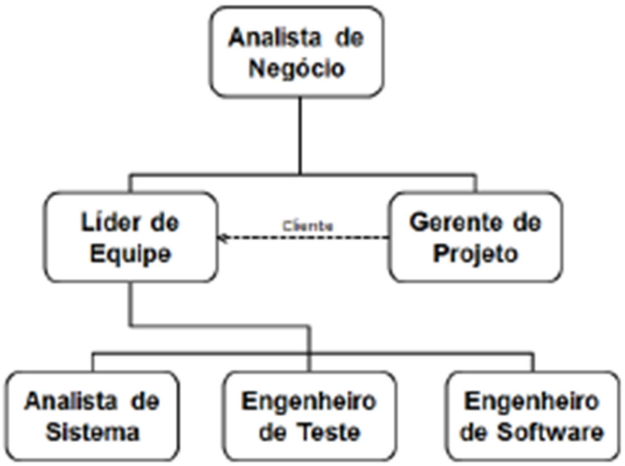
O iMobilis é um laboratório de pesquisa em computação, que desenvolve sistemas para dispositivos móveis, embarcados e Web por meio de práticas da Engenharia de *Software*. Tem como objetivos a capacitação especializada no desenvolvimento de projetos e a disseminação das tecnologias desenvolvidas e utilizadas pelos colaboradores. O laboratório se destaca por ser um ambiente de desenvolvimento e pesquisa dentro da universidade, onde os alunos têm a oportunidade de participar de projetos de pesquisa, além de ser um laboratório que distribui-se por várias disciplinas e pesquisas com ações de ensino.

Os projetos do iMobilis são acompanhados por um processo de desenvolvimento de *software* denominado BOPE, que possui características comuns aos métodos ágeis, mantendo um conjunto de atividades bem definidas que visam acelerar o desenvolvimento do *software*, buscando melhorias do processo e organização da equipe.

Segundo PEREIRA (2014), assim como o *Scrum*, o foco está na inspeção e validação dos produtos e artefatos pelo cliente. Por outro lado, assim com no XP, os produtos são desenvolvidos após os testes que são utilizados para verificá-los.

A Figura 25 apresenta os papéis que os membros do time de desenvolvimento podem desempenhar no processo BOPE. Conforme PEREIRA (2014), o líder da equipe é um desenvolvedor experiente, responsável por ajudar os desenvolvedores a realizarem suas atividades e verificar a aceitação das estórias de usuários realizadas pela equipe. Um desenvolvedor pode desempenhar três tipos de papéis: analista de sistemas, engenheiro de *software* e engenheiro de teste. Estes papéis são atribuídos segundo o perfil e experiência do desenvolvedor. O gerente de projeto acompanha todos os projetos, avalia o desempenho do processo de desenvolvimento, das equipes e dos produtos, e faz a gestão do risco dos projetos, considerando o cronograma, custo e qualidade.

Figura 25: Hierarquia nos papéis do BOPE



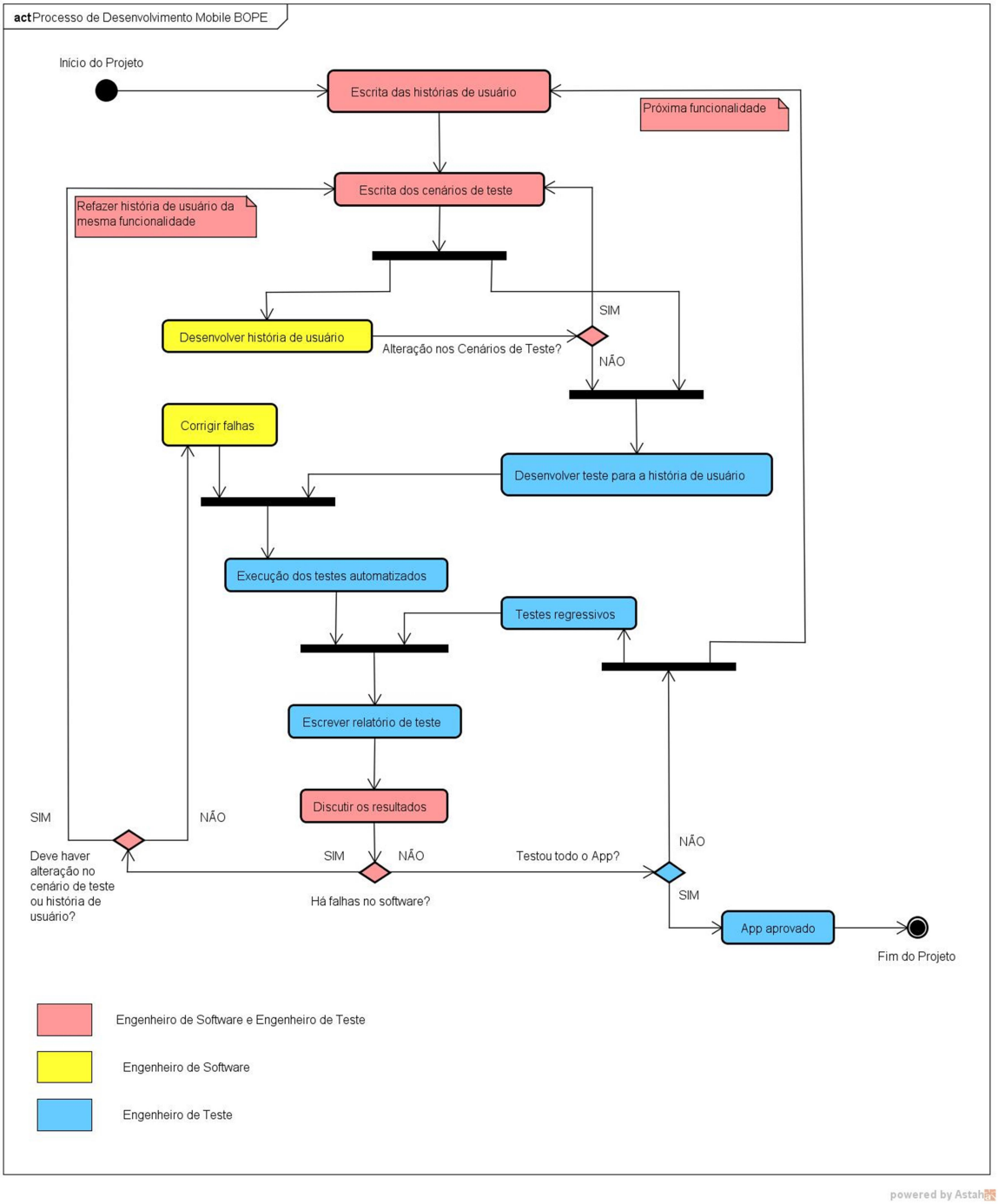
Fonte: Desenvolvendo *software* inovador em universidades públicas (PEREIRA, 2014)

Basicamente, sob perspectiva administrativa, todo projeto no iMobilis se inicia com uma reunião de planejamento, onde é estabelecido o escopo do projeto, ou seja, são definidos todos os seus entregáveis. A cada reunião de planejamento tem-se uma reunião de técnica e validação, momento em que as dúvidas técnicas são solucionadas e as atividades estabelecidas são validadas. No fim de cada *sprint*, são realizadas as reuniões de revisão e retrospectiva, onde o time de desenvolvedores demonstra as atividades como um todo e evidencia as lições aprendidas, destacando o que foi alcançado e o que pode ser melhorado para a próxima *sprint*.

Já sob a perspectiva de desenvolvimento, no laboratório não existe desenvolvimento sem teste ou teste sem desenvolvimento, eles são congruentes. O projeto se inicia com as histórias de usuário e cenários de teste, enquanto o engenheiro de *software* implementa a

história de usuário, o testador desenvolve os testes para as histórias do usuário. Os dois profissionais se comunicam o tempo todo, analisando os relatórios de teste, discutindo os resultados e implementando melhorias. Na Figura 26, o processo de teste idealizado pelo BOPE é ilustrado.

Figura 26: Processos de Teste *Mobile* Bope



Fonte: Adaptado de Laboratório iMobilis (2017)

4.1.2 Metodologia do estudo de caso

Foi selecionado o método de levantamento *(survey)* para conhecer o processo de teste do laboratório iMobilis. Um levantamento é um sistema para coletar informações de pessoas, para descrever, comparar ou explicar seus conhecimentos, atitudes e comportamentos. Um levantamento é muitas vezes uma investigação realizada em retrospectiva, quando, por exemplo, uma ferramenta ou técnica, está em uso por um tempo. O principal meio de coleta dos dados são as entrevistas ou os questionários (WOHLIN et al.,2012).

Dois questionários foram desenvolvidos e aplicados no iMobilis, a fim de obter informações necessárias para a pesquisa. Um questionário compreendeu a avaliação geral do processo de teste do laboratório, com o propósito de conhecer toda gestão e ciclo de vida do processo de teste utilizado atualmente, sendo direcionado ao profissional responsável pelo gerenciamento das atividades. O segundo questionário, também foi relacionado com a avaliação do processo de teste, porém direcionado aos profissionais de teste, ou seja, o objetivo aqui consistiu em obter conhecimento sobre a visão dos testadores sobre os processos de testes aplicados em diferentes projetos no laboratório. Ambos questionários estão disponíveis respectivamente nos apêndices A e B.

Algumas das estratégias de pesquisa podem ser informadas por critérios qualitativos ou quantitativos. Neste estudo, optou-se por critérios qualitativos. Segundo WOHLIN *et al.* (2012), o objetivo básico da análise qualitativa é extrair conclusões de dados, mantendo uma cadeia clara de evidências. Caracteriza-se por uma análise realizada em paralelo com a coleta de dados, onde novas descobertas são encontradas.

Os questionários foram compostos por perguntas objetivas. Para responder uma pergunta, era necessário ter conhecimento das definições de termos referentes ao teste de *software* e avaliar se a atividade associada era realizada no âmbito do processo de teste do laboratório.

4.1.3 Informações de apoio

A seguir serão apresentadas informações pertinentes aos questionários de avaliação do processo de teste do laboratório iMobilis, tanto a nível gerencial quanto a nível do projeto, foram baseadas no padrão ISO / IEC / IEEE / 29119-1 (2013):

- Gerenciamento de teste: planejamento, programação, estimativas, monitoramento, relatórios, controle e conclusão das atividades de teste.

- Nível de esforço: relacionado ao empenho, dedicação e interesse no processo de teste.
- Plano de teste: Descrição detalhada dos objetivos de teste a serem alcançados e os meios e cronograma para alcançá-los.
- Caso de teste: Conjunto de condições prévias de ensaio, insumos e resultados esperados, desenvolvidos para direcionar a execução de um item de teste para atender aos objetivos do teste, incluindo a implementação correta, a identificação de erros, verificação de qualidade e outras informações valiosas.
- Dados de teste: Dados criados ou selecionados para satisfazer os requisitos de entrada para a execução de um ou mais casos de teste, que podem ser definidos no plano de teste e no caso de teste.
- Ambiente de teste: Instalações, *hardware*, *software*, procedimentos e documentação destinados ou utilizados para realizar testes de *software*.
- Execução do teste: Processo de execução de um teste sobre o item de teste, produzindo resultado real.
- Teste de regressão: Testa as novas modificações em um item de teste ou em seu ambiente operacional, para identificar se as falhas de regressão ocorrem.
- Fases de teste: Instanciação específica de subprocesso de teste. São sinônimos de níveis de teste, como teste unitário, integração e aceitação.
- Cronograma de teste: Refere-se ao tempo acordado para a execução dos testes.
- Resultados de teste: Indicação da existência ou não de um caso de teste específico que foi aprovado ou rejeitado, ou seja, se o resultado real de saída do teste corresponde ao resultado esperado ou se foram observados desvios.
- Testes automatizados: A automatização dos testes requer o uso de *software* geralmente referidos como ferramentas de teste. Testes automatizados são muitas vezes direcionados principalmente a execução de testes de *scripts* em vez de ter testadores para executar testes manualmente.
- Processos de teste: Fornece informações sobre a qualidade de um produto de *software*, muitas vezes constituída por uma série de atividades, agrupadas em um ou mais subprocessos de teste.
- Estratégias de teste: Parte do plano de teste que descreve a abordagem de teste para um projeto de teste específico.
- Modelo de Maturidade: Abordagem que visa uma evolução bem definida no sentido de alcançar melhoria nos processos organizacionais, no contexto do trabalho, melhoria em processos de teste de *software*.

- Política de teste: Um documento que descreve o propósito, os objetivos e o escopo geral dos testes dentro de uma organização, expressa por que o teste é realizado e o que se espera alcançar.
- Planejamento de teste: Processo de gestão de teste utilizado para completar planejamento de teste e desenvolver planos de teste.
- Registro dos defeitos: Consiste em documentar os defeitos identificados.
- Entrada de teste: Condições para a realização do teste.
- Saída de teste: Condição esperada no fim do teste.
- Riscos de testes: Riscos relacionados à gestão de um projeto de teste, como por exemplo, falta de pessoas, recursos, prazos rígidos, mudanças de requisitos.
- Restrições quanto a forma de testar: Limitações relacionadas a um tipo de ferramenta específica ou requisitos.

A partir destas definições, é possível obter um maior entendimento sobre as questões abordadas na próxima seção, onde serão apresentados os detalhes dos resultados alcançados referente ao presente trabalho.

5 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Os questionários desenvolvidos foram aplicados no laboratório iMobilis, no dia 13/07/2017, e estão disponíveis nos Apêndices A e B. A principal contribuição dos questionários é a indicação de possíveis oportunidades para a melhoria do atual processo de teste do laboratório iMobilis. Os resultados obtidos na avaliação serão apresentados nesta seção, juntamente com a estratificação das questões abordadas.

5.1 Dados coletados no questionário de avaliação do processo de teste do laboratório iMobilis

O questionário do Apêndice A, composto por 14 questões relacionadas com a gestão de testes, foi respondido por um profissional envolvido com o nível gerencial do laboratório.

A preocupação inicial consistiu em identificar se a equipe de teste do laboratório tinha suas responsabilidades definidas e apoiadas. De acordo com o respondente, sim. Uma equipe de teste de *software* bem estruturada pode agregar diversos benefícios e valores aos produtos de *software*, bem como a garantia de uma melhor qualidade, reduzindo o índice de falhas por meio de testes concisos e completos.

Em seguida, foi necessário entender o nível de esforço dedicado pelo laboratório para as atividades de teste, para conhecer melhor a relação da empresa com o teste de *software*. Entre uma classificação de nenhum, muito pouco, pouco, mediano e alto, o respondente afirmou que o nível de esforço é mediano no iMobilis.

Posteriormente, constatou-se quais atividades de testes eram realizadas no laboratório, considerando as atividades essenciais no planejamento, preparação, especificação, execução e entrega, fases estas que compõe o ciclo de vida do teste de *software*. A seguir, a Tabela 1 apresenta os dados coletados:

É possível observar falhas na execução do ciclo de vida dos testes, na medida que os dados de teste não são preparados, casos de teste não são revistos e comparados. A preparação de dados de teste é extremamente importante, pois descreve o cumprimento de cada requisito de dados de teste. Recomenda-se a geração de um relatório para este fim, conforme definido na norma ISO / IEC / IEEE 29119-4.

Os casos de teste para um conjunto de teste podem ser selecionados com base nos riscos identificados, novos testes ou testes de regressão (ISO / IEC / IEEE 29119-2, 2013). Os casos de testes poderão e deverão ser alterados, de acordo com a liberação por parte da equipe de desenvolvimento dos módulos a serem testados. Aconselha-se a revisão de casos de testes a cada mudança significativa do código.

Tabela 1 – Respostas da questão 4 do questionário de avaliação de teste do Apêndice A

Atividades de Teste	**Aplicada no iMobilis**	**Não Aplicada no iMobilis**
Elaborar planos de teste	Sim	
Projetar casos de teste	Sim	
Preparar dados de teste		Não
Revisar casos de teste		Não
Documentar o processo de teste	Sim	
Executar programas com dados de teste	Sim	
Comparar resultados com casos de teste		Não
Preparar ambiente de teste	Sim	
Apresentar relatórios de teste	Sim	
Apresentar relatórios de tets	Sim	
Teste de regressão	Sim	

Fonte: Elaborado pelo autor

Outra questão muito importante foi conhecer como o desenvolvimento de *software* se correlacionava com os testes, se eram implementados pelos mesmos profissionais ou não, ou ainda, se eram integrados com outros processos. De acordo como o respondente, o grupo de teste trabalha dentro do desenvolvimento, reportando-se ao gerente ao Gerente de Projetos.

O teste de *software* ágil é caracterizado por ocorrer em todas as etapas do ciclo de vida de desenvolvimento de *software* e ser desempenhado por todos os membros do time (CAETANO, 2017). Neste contexto, uma sugestão para o laboratório iMobilis é não permitir que apenas um profissional seja ao mesmo tempo testador e desenvolvedor, ambas funções são importantes e possuem visões com os objetivos distintos dos testes realizados. Em testes ágeis, todo o time é responsável pelos testes, e cada membro contribui executando testes sob a sua perspectiva. Os desenvolvedores atuam na perspectiva do código, enquanto que os clientes avaliam a perspectiva da funcionalidade. O testador, complementa

os testes dos dois anteriores e colabora com abordagens mais especializadas. O testador se torna um papel fundamental na interação com os desenvolvedores, analistas de negócios e clientes.

O próximo passo da pesquisa, compreendeu em revelar se as fases de testes eram realizadas no laboratório. Seguir corretamente as fases de teste de *software* é fundamental para o sucesso ou fracasso de um projeto. As respostas apontaram que são realizados somente testes de sistema e testes de aceitação, testes de unidade e teste de integração não são implementados no iMobilis. O foco principal do teste ágil são os testes de unidade e de integração. O uso destas abordagens diminui o custo e a manutenção, pois os erros são encontrados mais cedo e aumenta a velocidade de entrega, pois não há o retorno para correção. Desta maneira, recomenda-se a implementação de testes unitários e de integração para melhoria do processo de teste do laboratório iMobilis.

O cronograma é uma das maiores preocupações no desenvolvimento de *software* e nas atividades de testes. Segundo o respondente, o laboratório não segue um cronograma de teste. Estabelecer um planejamento das atividades, detalhando o cronograma, com base no tempo, nos recursos, nos riscos e nas contingências para os testes é primordial para garantir a qualidade do produto. Por estas razões, indica-se a criação de um cronograma para todo processo de teste do laboratório iMobilis.

Após coletar as informações acima, procurou-se entender se os resultados dos testes executados, independentemente de terem encontrado erros ou não, eram registrados e disponibilizados para as partes interessadas, a fim de avaliar a abrangência do teste. Entre uma classificação de nunca, muito raramente, raramente, frequentemente e sempre, o respondente afirmou que os resultados dos testes são sempre registrados e encaminhados às partes interessadas.

Houve a necessidade de obter conhecimento se o laboratório aplicava testes automatizados, para ter a noção dos recursos disponíveis para os testes. A pesquisa apontou que a automação é aplicada em testes de aceitação, mas está em andamento em testes exploratórios. Testes automatizados são programas ou *scripts* simples que exercitam funcionalidades do sistema sendo testado e fazem verificações automáticas nos efeitos obtidos. A grande vantagem dos testes automatizados, é que todos os casos de teste podem ser facilmente e rapidamente repetidos a qualquer momento e com pouco esforço. No teste ágil, a automação tem grande enfoque, pois viabiliza curtos ciclos de entrega e faz parte de um ciclo de integração contínua, fornecendo resultados de modo contínuo. A automação libera a equipe para executar tarefas mais criativas ao invés de executar testes entediantes, repetitivos e manuais.

Outra questão abordada consistiu em saber se na visão do respondente, os processos de testes executados no laboratório eram suficientes para a garantia de qualidade do produto final ou não. Ele respondeu que não, que os testes atuais ajudam na melhoria de qualidade, mas novas ferramentas, técnicas e fases de teste podem ser adicionadas. Realmente, conforme destacado acima, muitos pontos de melhoria de processo de *software* ainda precisam ser implementados para que os testes realmente possam realmente garantir a qualidade do produto final, juntamente com os processos de desenvolvimento.

Foi importante também, questionar se havia diferença nos processos de testes de produtos internos, direcionados a comunidade acadêmica, e os produtos externos, como alguma empresa específica. O respondente declarou que os processos são os mesmos, independente do cliente. Contudo, nem todos os projetos os seguem. Neste ponto, é importante sugerir que todos os projetos sejam testados, para aumentar a confiabilidade dos *software*s e consequentemente a qualidade dos produtos finais do laboratório iMobilis.

Abordando diretamente a qualidade do produto, foi elaborada uma questão com o objetivo de identificar se além dos testes, outras ferramentas eram utilizadas para tratar a qualidade de *software*. O respondente informou que o laboratório está iniciando o uso do Sonar em alguns projetos, ferramenta que auxilia no gerenciamento de qualidade de código e projeto. Uma proposta aqui, seria intensificar o uso de inspeção no *software*. Atividades de inspeção de *software* têm se tornado uma alternativa importante para avaliar artefatos de *software* a fim de alcançar uma maior qualidade no processo de desenvolvimento.

Sabendo da importância dos modelos de maturidade e melhoria do processo de teste para melhorar o desempenho das organizações, procurou-se identificar se algum modelo era aplicado no laboratório. De acordo com os resultados da pesquisa, o iMobilis não possui nenhum modelo de maturidade ou de melhoria do processo de teste. Recomenda-se a utilização da série dos padrões ISO / IEC / IEEE 29119, para a melhoria do processo de teste no laboratório iMobilis. As normas ISO / IEC/ IEEE 29119 podem ser utilizadas com qualquer ciclo de desenvolvimento de *software* e em qualquer organização. Implementando estas normas, adotam-se os únicos padrões de testes acordados e reconhecidos internacionalmente, que irão prover a organização uma abordagem de alta qualidade para testes e que pode ser comunicada por todo o mundo.

Por fim, um espaço foi reservado um espaço para críticas, sugestões e opiniões do respondente, referente ao questionário respondido. Porém, nenhuma declaração foi registrada pelo respondente.

5.2 Dados coletados no questionário de avaliação do processo de teste do iMobilis direcionado para o profissional de teste

O questionário do Apêndice B, foi composto por 13 questões relacionadas com o processo de testes a nível dos projetos do laboratório iMobilis. O único critério para definir o perfil dos respondentes indicados para responder o instrumento de avaliação, foi o fato de serem membros da equipe de teste do laboratório iMobilis. A experiência com o processo empregado é de grande valia na indicação de evidências e associação de respostas para as questões abordadas.

A princípio, as perguntas iniciais foram relacionadas com a identificação do perfil do respondente. Dois participantes se disponibilizaram para responder o questionário. O respondente A, cumpre um papel mais voltado para o gerenciamento dos testes, está envolvido com o desenvolvimento de *software* há mais de cinco anos e sua experiência com teste de *software* é de um a três anos, tendo um nível de envolvimento considerado médio com o processo de teste no laboratório. Já o respondente B, está associado ao papel de testador. Possui uma experiência de um a três anos em desenvolvimento de *software* e em teste de *software*, e apresenta um alto nível de envolvimento com o processo de teste no iMobilis.

Na Tabela 2 são apresentadas as atividades relacionadas ao teste e os profissionais que as desenvolvem. É possível observar que o participante A está envolvido apenas com as questões de gerenciamento dos testes. Enquanto o participante B, realiza todas as atividades, exceto o treinamento da equipe de teste.

Tabela 2 – Respostas da questão 5 do questionário de avaliação de teste do Apêndice B

Atividades de Teste	Participante A	Participante B
Política de teste	Sim	Sim
Planejamento de teste	Sim	Sim
Projeto de casos de teste	Sim	Sim
Especificação dos procedimentos de teste	Sim	Sim
Execução do teste	Não	Sim
Registro dos defeitos encontrados	Não	Sim
Preparação do ambiente de teste	Não	Sim
Auditorias	Não	Sim
Acompanhamento do projeto de teste	Sim	Sim
Treinamento de equipe de teste	Sim	Não
Automação de teste	Não	Sim
Documentação de teste	Não	Sim
Melhoria do processo de teste	Sim	Sim
Avaliação da ferramenta de teste	Não	Sim

Fonte: Elaborado pelo autor

Outras questões mais específicas de algumas determinas atividades, foram abordadas para ter conhecimento do processo de teste nos projetos do laboratório, estão dispostas na Tabela 3. Com os resultados da pesquisa foi possível identificar que os riscos não são revisados frequentemente pelos profissionais. De acordo com a ISO / IEC / IEEE 29119-1, durante os testes atividades de monitoramento devem realizadas para assegurar que o teste esteja progredindo conforme o planejado e para garantir que os riscos sejam tratados de maneira adequada. Riscos devem ser acompanhados, controlados e mitigados. Assim, uma sugestão de melhoria seria a realização de testes baseados em risco, que garante que os riscos com a maior prioridade recebam a maior atenção durante o teste.

Tabela 3 – Respostas das questões específcas do questionário (6 a 12)

Atividades de Teste	Participante A	Participante B
O planejamento de teste é uma atividade no processo?	Sim	Sim
Existe uma documentação de casos de teste?	Sim	Sim
São definidos os conjuntos de entradas e saídas para os testes?	Sim	Sim
Todos os riscos são frequentemente revisados?	Não	Não
São listadas restrições com relação ao modo de testar?	Não	Não
As atividades de teste são revistas com frequencia?	Sim	Sim
Os dados da execução do teste são registrados?	Frequentemente	Frequentemente

Fonte: Elaborado pelo autor

Outro ponto apresentado na Tabela 3 é que os profissionais não listam restrições com relação ao modo de testar. Limitações relacionadas a um tipo de ferramenta específica ou requisitos podem influenciar na realização e resultados dos testes. Aconselha-se efetuar análises de todo ambiente de teste e recursos antes de iniciar as atividades, antecipando-se aos possíveis riscos.

Finalmente, foi reservado um espaço para críticas, sugestões e opiniões dos respondentes, referente ao questionário respondido. Somente o participante A registrou sugestões, indicando que seria relevante questionar diretamente sobre o papel do respondente e quantas pessoas participam da equipe de teste.

Neste estudo observou-se que o laboratório iMobilis segue uma abordagem de metodologia híbrida na gestão dos processos de desenvolvimento de *software*. O gerenciamento de projetos é baseado no Guia PMBOK, contudo, também é aplicado práticas ágeis provenientes dos modelos *Scrum* e XP. É importante encontrar um ponto de

equilíbrio para identificar em cada uma das abordagens quais são os processos, ferramentas e técnicas que melhor se adaptam a necessidade de projetos do laboratório, e a partir daí sim, criar uma metodologia própria.

6 CONSIDERAÇÕES FINAIS

Manter a qualidade no desenvolvimento de *software* é um desafio enfrentado por muitas organizações. Diversos problemas impactam na qualidade de *software*, como a não execução de testes de *software* de maneira correta e a não existência de uma gestão de processos de desenvolvimento em níveis adequados.

Para obter a garantia de qualidade de um *software* é fundamental investir nos testes. Estes devem estar presentes durante todo o processo de desenvolvimento do *software* em questão, identificando possíveis falhas e defeitos, o que evita um custo excedente e desnecessário ao projeto e ao cliente.

No decorrer do trabalho foi possível perceber a importância da existência de normas para que se tenha uma padronização no modo com que são realizados os processos, os documentos, as técnicas, facilitando a comunicação tanto entre as equipes de uma organização, quanto entre organizações diferentes. O conjunto de normas ISO / IEC / IEEE 29119, surgiu como resposta aos conflitos e lacunas que existiam referentes aos padrões relacionados ao teste de *software*. Os consumidores de serviços de testes de *software* e os próprios testadores, não tinham uma única fonte de informação sobre boas práticas de teste.

A série ISO / IEC / IEEE 29119 é um conjunto integrado de padrões internacionais de testes de *software* que fornecem uma cobertura muito mais ampla da disciplina de testes, e também uma solução pragmática para ajudar as organizações e testadores (REID, 2014). Essas normas podem ser utilizadas com qualquer ciclo de desenvolvimento de *software* e em qualquer organização.

Por outro lado, algo importante a ser considerado para a garantia de qualidade, é o processo de desenvolvimento de *software*. Os métodos ágeis, mitigam os riscos e desafios associados a uma alta incidência de defeitos por meio de testes. Enquanto nas metodologias tradicionais o teste é uma atividade realizada ao final do desenvolvimento por uma equipe independente, nas metodologias ágeis o teste é uma atividade colaborativa feita por todos membros do time durante todo o processo de desenvolvimento do *software*.

O teste de *software* é o pilar de sustentação que permite a implementação de muitos princípios de desenvolvimento ágil. Os testes são o único meio de demonstrar se o *software* atende às necessidades do cliente, eles aumentam a confiança da equipe para realizar mudanças sem medo de causar efeitos contrários no software. Testes apresentam as falhas existentes tanto no código quanto no processo de trabalho, com base nas lições aprendidas na identificação da falha, a equipe é capaz de otimizar o seu comportamento.

O estudo de caso realizado no laboratório iMobilis contribuiu para a aplicação prática do embasamento teórico referente a integração dos padrões ISO / IEC / IEEE 29119 e os métodos ágeis de desenvolvimento de *software*. Os resultados das informações coletadas na pesquisa relacionada ao atual processo de testes aplicado no laboratório, evidenciaram que o mesmo apresenta uma estruturação técnica quanto ao seu processo de teste. Porém, existem vários pontos de atenção que precisam ser ajustados para buscar a melhoria do processo de teste. As sugestões de melhorias foram fundamentadas nos padrões e nos métodos ágeis.

Uma oportunidade para a expansão futura do trabalho seria verificar a possibilidade de aplicação das melhorias sugeridas no processo de teste, uma vez que o laboratório iMobilis é um ambiente de pesquisa e sendo assim, propício a fazer novos estudos, tornando-se capaz de oferecer ações a serem tomadas em resposta aos levantamentos apresentados.

Por fim, foi possível compreender que devido ao mercado cada vez mais competitivo, é imprescindível que as organizações estejam sempre atualizadas, adotando novas maneiras de gerenciamento e inovação, para que consigam entregar projetos eficazes e rápidos. Por estas razões, os padrões da série ISO/IEC/IEEE 29119 podem ser integrados em diferentes métodos ágeis, para que melhores resultados sejam alcançados, maximizando a qualidade do produto de *software* desenvolvido.

REFERÊNCIAS

AFZAL, W; ALONE, S; GLOCKSIEN, K; TORKAR, R. ***Software test process improvement approaches: A systematic literature review and an industrial case study.*** *The Journal of Systems and Software*, 111, 2016.

BECK, K. ***Extreme Programming Explained: Embrace Change***. 2. Ed. Boston: Addison Wesley, 1999.

BECK, K; et al. ***Manifesto for Agile Software Development, 2001.*** Disponível em: <https://goo.gl/ruvK>. Acesso em: 21/07/2017.

CAETANO, C. ***Testes Ágeis.*** 2017. Disponível em: <https://goo.gl/qVy34T>. Acesso em: 10/08/2017.

DELAMARO, M.E.; MALDONADO, J.C.; JINO, M. **Introdução ao Teste de *Software***. Elsevier Editora, 2007.

HWANG, S; ***Software Test Capability Improvement through a Lightweight Test Process.*** *Journal of Security Engineering,* 2012.

ISO/IEC 25010. ***Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models.*** Geneva: International Organization for Standardization, 2011.

ISO/IEC/IEEE 29119-1. ***Software and systems engineering - Software testing - Part 1: Concepts and definitions.*** Geneva, Switzerland: International Organization for Standardization, 2013.

ISO/IEC/IEEE 29119-2. ***Software and systems engineering - Software testing - Part 2: Test processes.*** Geneva, Switzerland: International Organization for Standardization, 2013.

ISO/IEC/IEEE 29119-3. ***Software and systems engineering - Software testing - Part 3: Test documentation.*** Geneva, Switzerland: International Organization for Standardization, 2013.

ISO/IEC/IEEE 29119-4. ***Software and systems engineering - Software testing - Part 4: Test techniques.*** Geneva, Switzerland: International Organization for Standardization, 2015.

ISO/IEC/IEEE 29119-5. ***Software and systems engineering - Software testing - Part 5: Keyword-Driven Testing.*** Geneva, Switzerland: International Organization for Standardization, 2016.

MEYER, B; ***Agile The Good, the Hype and the Ugly.*** Springer, 2014.

PEREIRA, Igor Muzetti. ***Desenvolvendo software inovador em universidadespúblicas: Adaptando processos ágeis para a realidade de laboratórios depesquisa e desenvolvimento.*** 2014. 63 f. Mestrado (Ciência da Computação) – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de OuroPreto, Ouro Preto. 2014.

PEZZÈ, M.; YOUNG, M.; ***Teste e Análise de Software.*** Porto Alegre: Bookman, 2008.

PMI. ***A Guide to the Project Management Body of Knowledge (PMBOK Guide).*** Pennsylvania: PMI,. 2008.

PRESSMAN, R. S***. Engenharia de Software Uma Abordagem Profissional.*** 7. Ed. 2011.

REID, S. ***ISO/IEC/IEEE 29119 Software Testing Standard***. 2014. Disponível em: <http://www.*software*testingstandard.org>. Acesso em 03/07/2017.

TMMi FOUNDATION. ***Teste de integração modelo de maturidade (TMMi)*****.** 2010. Disponível em:<https://goo.gl/5HBQh7>. Acesso em: 23/07/2017.

TPI AUTOMOTIVE. **Teste Process Improvement**. 2004. Disponível em: <https://goo.gl/L1u9MQ> Acesso em: 23/07/2017.

SOMMERVILLE, I. ***Engenharia de Software.*** 9. Ed. Pearson Education, 2011.

WOHLIN, C., RUNESON, P. HÖST, M., OHLSSON, M. C., REGNELL, B., and WESSLÉN, A***. Experimentation in Software Engineering***. Berlim: Springer-Verlag, 2012.

APÊNDICE A – Questionário de Avaliação do Processo de Teste do Laboratório iMobilis

Objetivo: A principal contribuição do questionário é a indicação de possíveis oportunidades para a melhoria do atual processo de teste do laboratório iMobilis.

1. Identificação do Respondedor:

O nome do respondente não será divulgado neste trabalho, será mantido o total sigilo sobre sua identidade.

Nome (Opcional): _____

Formação Acadêmica: _____ Cargo: _____

Email: _____ Data: __/__/____

2. A equipe de teste tem suas responsabilidades claramente definidas e apoiadas? Justifique sua resposta.

☐ Sim ☐ Não

_____

_____

_____

3. Qual o nível de esforço dedicado pelo laboratório para as atividades de teste?

☐ Nenhum ☐ Muito pouco ☐ Pouco ☐ Mediano ☐ Alto

4.Quais as principais atividades referentes aos testes são realizadas no laboratório? (Permitido marcar mais de uma opção).

☐ Elaborar plano de teste
☐ Executar programas com dados de teste
☐ Projetar casos de teste
☐ Comparar resultados dos casos de teste
☐ Preparar dados de teste
☐ Preparar ambiente de teste
☐ Revisar casos de teste
☐ Apresentar relatórios de teste
☐ Documentar o processo de teste
☐ Teste de regressão
☐ Outras atividades? Quais? _____

_____

5. Como o grupo de teste está organizado no laboratório? (Permitido marcar somente uma).

☐ Os desenvolvedores fazem o teste.

☐ Grupo de teste dentro do desenvolvimento reportando-se ao Gerente de Projetos.

☐ Grupo de teste separado reportando-se ao Gerente de Teste.

☐ Parte do grupo de Garantia da Qualidade.

☐ Outros (Por favor, especifique): _____

6. Quais fases de testes são realizadas no laboratório? (Permitido marcar mais de uma)

☐ Teste de unidade ☐ Teste de Integração ☐ Teste de Sistema ☐ Testes de Aceitação

7. O cronograma de teste é monitorado no Laboratório?

☐ Sim ☐ Não ☐ Não existe um cronograma

8. É fornecido algum relatório com as informações sobre os resultados dos testes para os interessados?

☐ Nunca ☐ Muito Raramente ☐ Raramente ☐ Frequentemente ☐ Sempre

9. Em quais fases são utilizados os testes automatizados? Quais são as ferramentas empregadas?

_____

_____

_____

10. Em geral, os testes são suficientes para garantir a qualidade nos produtos finais? Justifique.

_____

_____

_____

11. Existe alguma diferença nos processos de testes realizados em produtos direcionados aos clientes internos (Ex: Comunidade Acadêmica) e externos (Ex:Vale)?

_____

_____

_____

12. São usadas outras técnicas ou ferramentas para tratar a qualidade de *software*, além dos testes? Quais?

_____

_____

_____

13. O laborotário utiliza algum modelo de maturidade ou de melhoria de processo de teste (ex: CMMi, TPI, ISO IEC IEEE 29119)? Qual?

_____

_____

_____

14. Existe algum assunto que não foi abordado e que você acha relevante para esta pesquisa? Por favor, especifique.

_____

_____

_____

Obrigada pela participação!

APÊNDICE B – Questionário de Avaliação do Processo de Teste do Imobilis para o profissional de teste

Objetivo: A principal contribuição do questionário é a indicação de possíveis oportunidades para a melhoria do atual processo de teste do laboratório iMobilis.

1. Identificação do Respondedor:

O nome do respondente não será divulgado neste trabalho, será mantido o total sigilo sobre sua identidade.

Nome (Opcional): _____

Formação Acadêmica: _____ Cargo: _____

Email: _____ Data: __/__/____

2. Qual é o seu tempo de experiência em desenvolvimento de *software*?

☐ Até um ano ☐ Entre um a três anos ☐ Entre três a cinco anos ☐ Mais de cinco anos

3. Qual é o seu tempo de experiência em teste de *software*?

☐ Até um ano ☐ Entre um a três ano ☐ Entre três a cinco ano ☐ Mais de cinco anos

4. Qual o nível de seu envolvimento com o planejamento e execução das atividades do processo de teste no laboratório iMobilis?

☐ Nenhum ☐ Muito pouco ☐ Pouco ☐ Médio ☐ Alto

5. Quais atividades relacionadas ao teste você está efetivamente envolvido? (Permitido marcar mais de uma)

☐ Política de teste
☐ Planejamento de teste
☐ Projeto de casos deteste
☐ Especificação dos procedimentos de teste
☐ Execução do teste
☐ Registro dos defeitos encontrados
☐ Preparação do Ambiente de Teste
☐ Treinamento da equipe de teste
☐ Automação do teste
☐ Documentação de teste
☐ Melhoria do processo de teste
☐ Avaliação de ferramentas de teste
☐ Nenhuma
☐ Outros (Por favor, especifique)

_____

6. O Planejamento de Testes é uma atividade no processo?

☐ Sim ☐ Não

7. Existe uma documentação de casos de teste?

☐ Sim ☐ Não

8.São definidos os conjuntos de entradas e saídas para os testes?

☐ Sim ☐ Não

9. Todos os riscos são freqüentemente revisados?

☐ Sim ☐ Não

10. São listadas as restrições com relação ao modo de testar?

☐ Sim ☐ Não

11.As atividades de teste são revistas com frequência?

☐ Sim ☐ Não

12. Os dados da execução do teste são registrados?

☐ Nunca ☐ Raramente ☐ Frequentemente ☐ Sempre

13. Existe algum assunto que não foi abordado e que você acha relevante para esta pesquisa? Por favor, especifique.

_____

_____

_____

Obrigada pela participação!

APÊNDICE C – Listagem dos Artigos Selecionados para Estudo

Tabela 4 – Listagem de artigos selecionados para estudo

Nome do Artigo	Ano de publicação	Link do artigo
Software test process improvement approaches: A systematic literature review and an industrial case study	2016	https://goo.gl/Xbm7CP
Diagnostic of *Software* Using Testing Time and Testing Coverage	2016	https://goo.gl/ouGVty
Maturity Level Assessment in *Software* Testing in Small and Medium-Sized Enterprises of the State of Goiás	2015	https://goo.gl/CCLfKW
Risk orientation in *software* testing processes of small and medium enterprises: an exploratory and comparative study	2015	https://goo.gl/knDXEv
On the Effects of Programming and Testing Skills on External Quality and Productivity in a Test-Driven Development Context	2015	https://goo.gl/5c4KZ3
Towards a Test Automation Improvement Model (TAIM)	2014	https://goo.gl/BhNMJV
Supporting Regression Test Scoping with Visual Analytics	2014	https://goo.gl/Sn1uSG
Test Process Improvement with Documentation Driven Integration Testing	2014	https://goo.gl/cLsmkv
Identifying Process Improvement Targets in Test Processes: A Case Study	2013	https://goo.gl/m2Tyn6
Software Testing Process Performance Improvement using Service-Based Testing Support	2012	https://goo.gl/Vg81hz

Fonte: Elaborado pelo autor

APÊNDICE D – Listagem dos Artigos Selecionados para Estudo

Tabela 5 – Listagem de artigos selecionados para estudo

Nome do Artigo	Ano de publicação	Link do artigo
Hybriding CMMI and Requirement Engineering Maturity & Capability Models	2012	https://goo.gl/RA34o9
Technical Debt in Test Automation	2012	https://goo.gl/Du8Ppi
Estimating the Return on Investment of Defect Taxonomy Supported System Testing in Industrial Projects	2012	https://goo.gl/F6cSLx
Software Test Capability Improvement through a Lightweight Test Process	2012	https://goo.gl/gmgbic
How Test Organizations Adopt New Testing Practices and Methods?	2011	https://goo.gl/5smXpK
Model Based Testing - Executable State Diagrams	2011	https://goo.gl/r5YzBN
Elaborating *Software* Test Processes and Strategies	2010	https://goo.gl/rn5dUp
Software Testing Optimization by Advanced Quantitative Defect Management	2010	https://goo.gl/ffK1hz
The definiton of a testing process to small-sized companies: the Brazilian scenario	2010	https://goo.gl/tdVTRi
Software Test Factory (A proposal of a process model to create a Test Factory)	2010	https://goo.gl/i4KPTC

Fonte: Elaborado pelo autor