Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# Ferramenta de gestão de riscos aplicada a ambientes de desenvolvimento de software com foco na garantia da qualidade do produto

Ana Cristina Fernandes Lima

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientadora
Profa. Dra. Simone Borges Simão Monteiro

Brasília
2019

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Ff

Fernandes Lima, Ana Cristina
Ferramenta de gestão de riscos aplicada a ambientes de desenvolvimento de software com foco na garantia da qualidade do produto / Ana Cristina Fernandes Lima; orientador Simone Borges Simão Monteiro. -- Brasília, 2019.
194 p.

Tese (Doutorado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2019.

1. Engenharia de Software. 2. Qualidade de Software. 3. Testes de Software. 4. Gestão de Risco. 5. Teste de Software Baseado em Risco. I. Simão Monteiro, Simone Borges , orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# Ferramenta de gestão de riscos aplicada a ambientes de desenvolvimento de software com foco na garantia da qualidade do produto

Ana Cristina Fernandes Lima

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Profa. Dra. Simone Borges Simão Monteiro (Orientadora)
CIC/UnB

Prof. Dr. Edgard Costa Oliveira
Universidade de Brasília

Prof. Dr. Sérgio da Costa Côrtes
Instituto de Ensino Superior de Brasília

Profa. Dra. Aletéia Patrícia Favacho de Araújo Von Paumgartten
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 31 de julho de 2019

# Dedicatória

Este trabalho é dedicado a Deus, que tem sido meu fiel ajudador, dando-me vida e paz por meio de sua infinita graça e misericórdia e à minha maravilhosa e extraordinária família (Marido Marcello, filho Luiz Henrique e Pai Luis Alves), que me apoiam em todos os momentos da vida. Sem eles eu não conseguiria. É dedicado também a todos que em algum momento disseram que eu não conseguiria, pois esses também me deram força para buscar ainda mais este objetivo.

Dedico a minha orientadora Profa. Dra. Simone Borges Simão Monteiro, por todos os ensinamentos e contribuições valiosas, e todo conhecimento adquirido e experiências únicas passadas pelo tempo do mestrado. Destaco todo a garra, profissionalismo e dedicação empregada nas atividades do mestrado e ao curso de Engenharia de Produção que fazem toda a diferença.

# Agradecimentos

Ao chegar à conclusão desta obra gostaria de agradecer a todos que me ajudaram a alcançar esse importante objetivo. A Deus, por me conceder vida, saúde e capacidade física, emocional e intelectual. Ao meu filho, por todo o apoio desde o dia que nasceu, veio para mudar minha vida e me dar apoio em todos esses momentos, e por sua compreensão quanto às minhas ausências. Ao meu lindo marido, por todo o apoio emocional, incentivo e companheirismo em todos os momentos, como também por sua generosa compreensão quanto à minha ausência durante a realização deste trabalho. A meu pai Luis Alves Fernandes Filho, que sempre acreditou em mim, e foi um dos maiores incentivadores desde a minha graduação, sempre acreditando no meu potencial. A minha avó Maria Gilda Ferreira Fer-nandes e Gleyde Maria, que sempre me apoiaram e me ajudaram a cuidar do meu filho nos momentos de ausência, e sempre intercedeu junto a Deus pela benção para a realização dos meus sonhos. A minha família querida que sempre está comigo em todas, me apoiando e me dando força, Rita de Cassia, Thiago, Felipe, Patricia e Marco. Às minhas amigas Gabriela Rodrigues e Elisabete Santos que sempre me apoiaram nos estudos e na vida pessoal para me dar força a alcançar esse objetivo. A minha mãe Luziana Leite, irmã Monique Fernandes e irmão Pedro Victor por todo apoio dado nesta caminhada. A RSI Informática que me autorizou a dedicar parte do meu tempo de expediente para participação nas aulas e na elaboração da pesquisa. Destaque-se neste sentido José Roberto Murillo Zamora Sócio da empresa, Eduardo Medeiros Diretor de Governo e Antonio Zamora Comitê da presidência da empresa. A minha orientadora Prof.a Dr.a Simone Borges Simão Monteiro, por toda sua excelência e comprometimento na orientação, que se destaca na qualidade de ensino junto ao mestrado do PPCA e por todos incentivos de publicação que foi dado ao longo da caminhada e todo aprendizado. A Profa. Dra. Ana Carla Bittencourt pela contribuição dada na qualificação e na publicação de artigo. À CAPES pela disponibilidade do Portal de Periódicos à Universidade que permitiu realizar pesquisas de alto impacto e ao Gustavo Portella, coordenador Geral de Informática por todo incentivo. Ao Prof. Sérgio Côrtes pela honra de participar na minha banca e por todos os ensinamentos, frente a diretoria da CAPES. Finalmente, mas não menos importante, a todo corpo docente do Mestrado.

# Resumo

Com o aumento da quantidade de softwares, cresce também o número de falhas, com entregas de produtos com erros, ou que não atendam às expectativas dos clientes, gerando impacto, retrabalho e insatisfação. A busca pela qualidade de software é um fator crítico que tem crescido com a complexidade das tecnologias e a competitividade entre as empresas. A atividade de teste, exerce um importante papel dentro da garantia da qualidade de software, assegurando a identificação de defeitos, com intuito de reduzir os riscos associados ao produto de software, buscando uma qualidade aceitável. Esse trabalho visa contribuir no provimento de uma ferramenta automatizada, que por meio da gestão de risco, possa auxiliar a garantia da qualidade do produto de software, através do controle de testes específicos em suas fases de desenvolvimento. A ferramenta foi desenvolvida a partir de pesquisas na literatura, e levantamento das necessidades, compreendendo técnicas de coleta de dados por meio de pesquisa não estruturada, *benchmarking*, entrevistas e extração em base de dados. Sua concepção foi embasada nas fases de gestão de risco da ISO31000 [1], considerando os conceitos da *Risk Based Test (RBT)*, incorporadas ás concepções de qualidade da ISO25010[2], e foi validada por meio de estudos de casos qualitativos, permitindo a identificação, análise e monitoramento dos riscos dos produtos de software, por meio de suas funcionalidades, subsidiando assim a melhoria da qualidade dos produtos. As validações, possibilitaram a validação da ferramenta proposta, por meio de contextos organizacionais distintos e os resultados apresentaram uma grande efetividade, permitindo a identificação de uma série defeitos, por meio da realização de testes indicados na ferramenta, que foram providos através da identificação dos riscos relacionados as características de qualidade da ISO25010[2], providos por meio de análise de risco qualitativa nas funcionalidades do software, possibilitando assim, a redução dos riscos associados a qualidade do software, antes de serem disponibilizados a seu ambiente produtivo.

**Palavras-chave:** Engenharia de Software, Qualidade de Software, Testes de Software, Gestão de Risco

# Abstract

As the number of software increases, so does the number of failures, product deliveries that are out of order or failing to meet customer expectations, causing rework and dissatisfaction. The search for software quality is a critical factor that has grown side by side with technology complexities and competition among companies. The testing activity has an important role in software quality assurance aiming to guarantee the quality of the software to enable defect identification, in order to reduce the risks associated with the software product, aiming towards acceptable quality. This work intends to contribute to this matter by providing an automated tool that will help garantee acceptable software quality, using risk management and test control throughout developmental stages. The tool was developed from literature research and necessity assessments, Non-Structured Data Collection Techniques, benchmarking, interviews and database extraction. The tool was developed based on ISO31000 2.12 risk management phases, considering the concepts of the Risk Based Test (RBT), incorporated into the [2] quality conception, and has been validated through qualitative case studies that allowed the identification, analysis and monitoring of software product risks by their functionalities, validating the improvement in software product overall quality. Case studies allowed the proposed tool validation through distinct organizational contexts and the results presented proof of effectiveness, enabling the identification of a series of defects, using the tests provided within the tool, made possible by the identification of risks related to the quality characteristics of [2] associated to a qualitative risk analysis in the software functionalities, thus enabling the reduction of the Risks associated with software quality before making it available to production environment.

**Keywords:** Software Engineering, Quality Software, Risk Management, Software Test

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas e Siglas

**APEX** Oracle Application Express.

**BI** Business Intelligence.

**BRB** Banco de Brasília.

**CIA** Central Intelligence Agency.

**CMMI** Capability Maturity Model Integrated.

**CPD** Centro de Processamento de Dados.

**HTML** Hypertext Markup Language.

**IBM** International Business Machines.

**ISO** International Organization for Standardization.

**ISTQB** International Software Testing Qualifications Board.

**MER** Modelo Entidade Relacionamento.

**NASA** National Aeronautics and Space Administration.

**PBL** Abordagem Baseada em Pprojeto.

**PL/SQL** Procedural Language/Structured Query Language.

**PO** Product Owner.

**PUMA** Plataforma Unificada de Metodologia Ativa.

**RBT** Risk Based Test.

**RSI** RSI Informática.

**RUP** Rational Unified Process.

**SEI** Software Engineering Institute.

**SQA** Software Quality Assurance.

**TI** Tecnologia da Informação.

**UML** Unified Modeling Language.

**XP** Extreme Programming.

# Capítulo 1

# Introdução

Neste capítulo serão apresentadas as informações principais que contextualizam a problemática a ser tratada, a motivação, a definição do problema e o objetivo geral e os objetivos específicos da pesquisa.

## 1.1 Contextualização

Os investimentos globais em Tecnologia da Informação (TI), segundo estimativa realizada pela consultoria *Gartner*, terão um acréscimo de 1,1% em 2019 em relação ao ano de 2018, chegando a 3,79 trilhões. O segmento que apresentará o maior crescimento nos próximos anos, será a área de software, contando com uma previsão crescimento de 7,1% em 2018 e 8,2% em 2019, totalizando 462 bilhões[3].

Nesse cenário, há grande necessidade de focar na qualidade do software. Segundo Mecenas e Oliveira [4], a produção de software deixou de ser, há algum tempo, uma atividade baseada apenas na intuição ou na experiência dos desenvolvedores.

O processo de desenvolvimento de software tem sido objeto de inúmeros estudos, há mais de três décadas [4], numa tentativa de derivar modelos, processos e ferramentas que possibilitem o gerenciamento das fases de produção e assegurem que os produtos tenham a qualidade desejada pelos consumidores.

De maneira geral, as empresas, têm despertado para a importância da atividade de teste de software, como forma de melhorar a qualidade dos seus produtos e manterem-se competitivas no mercado. Conduto, a complexidade das tecnologias utilizadas e dos produtos de software produzidos tem crescido, tornando-se indispensável a utilização de processos, técnicas, ferramentas e métodos, que permitam a realização de teste de software de maneira sistemática e com fundamentação teórica, com o propósito de aumentar a qualidade do software, com o menor custo possível [5]. Desta forma, o processo de teste

exerce um importante papel dentro da garantia de qualidade de software, assegurando que os requisitos satisfaçam as necessidades das partes envolvidas.

Por sua vez, os modelos de qualidade permitem avaliar softwares de acordo com diferentes aspectos e comumente representam as características desejáveis a um software em uma estrutura hierárquica [6].

Neste sentido, a atividade de teste tem como objetivo encontrar defeitos, reduzir os riscos associados a um sistema e identificar o máximo de problemas, efetivando a qualidade dos produtos desenvolvidos. No entanto, esta atividade é difícil e custosa de ser realizada, uma vez que, o domínio de entradas e saídas de um software são diversos, bem como variadas as possibilidades e caminhos possíveis a serem testados dentro de um sistema. Além disso, segundo Kaner[7], a atividade de teste é bastante cara, chegando a custar até 45% do valor inicial de um produto em desenvolvimento.

Entretanto, esta visão está sendo substituída por uma outra que privilegia a qualidade e como consequência, enfatiza o maior esforço no processo entendendo que testar é um investimento em qualidade e, que este produz um retorno positivo contribuindo decisivamente para o sucesso do projeto [8].

Por outro lado, a gestão adequada de riscos em projetos de software, tornam-se premissas para que esses se desenvolvam de forma adequada e gerem maior valor possível às partes interessadas a uma qualidade aceitável. Para que isso ocorra é importante o apoio de processos, ferramentas e técnicas para a realização de atividades contínuas de identificação, análise, avaliação e monitoramento dos riscos do produto de software.

A atividade de teste de software, por sua vez, dá suporte a garantia da qualidade, através de projeção, execução de teste, identificação de defeitos e análise dos resultados, permitindo a redução de riscos e falhas em seu ambiente produtivo[9].

Atualmente, algumas literaturas tratam sobre a gestão de riscos na qualidade do software [10, 11, 12, 13], e oferecem conjuntos de processos no sentido de facilitar o gerenciamento de riscos de um produto de software, tais como a normas *International Organization for Standardization (ISO)* 310000 [1], Qualidade de software [2] e o Teste de Software Baseado em Risco[14], dentre outros que serão apresentados neste trabalho, os quais forneceram embasamento para o desenvolvimento dessa pesquisa e serão apresentados no Capítulo 2 deste trabalho.

## 1.2 Definição do Problema

O processo de software envolve muitas fases em seu ciclo de vida de desenvolvimento. É necessário ter cautela, planejamento e apoio de ferramentas no processo, caso contrário,

erros graves podem ocorrer durante as várias etapas, inclusive quando o sistema entra em seu ambiente produtivo causando impactos, que em alguns casos podem ser imensuráveis.

Um relatório recentemente publicado pela empresa *Tricentis*[15], apresenta um estudo com a investigação de 606 falhas de software em 314 empresas. A análise mostra que os *bugs*, impactaram metade da população mundial 3,7 bilhões de pessoas e 1,7 trilhões em ativos das empresas.

Neste contexto, o processo de teste é a fase do desenvolvimento do software que tem como objetivo encontrar e remover defeitos e avaliar o grau de qualidade de um produto e dos seus componentes de acordo com as necessidades de cada cliente ou usuário, minimizando os riscos de erros no ambiente produtivo, buscando uma qualidade aceitável e agregando valor ao produto [13, 7].

De acordo com a regra 10 de Myers [16], quanto mais tarde um erro é encontrado, maior é o custo para corrigi-lo. Esse custo de correção cresce exponencialmente na escala de 10 vezes para cada estágio em que o projeto de software avança em suas fases de desenvolvimento.

Os modelos de qualidade, por sua vez permitem, através de normas internacionais, a avaliação dos softwares, com base em características de qualidade que podem ser verificadas por testes de softwares[17]. Por outro lado, infelizmente, o teste ainda é visto, por alguns, como uma etapa que tem seu início ao final do processo de desenvolvimento[18].

Desta forma, à medida que o tamanho e a complexidade dos softwares crescem, aumenta a necessidade da utilização de metodologias para gerenciamento de risco [19].

Considerando os cenários apresentados, existem abordagens e ferramentas que trabalham com riscos relacionados à engenharia de produto ou riscos técnicos, e consiste em um conjunto de atividades que favorecem a identificação de fatores de riscos associados aos requisitos do software. Uma vez identificados, os riscos são priorizados de acordo com a sua probabilidade de ocorrência e impacto, objetivando a redução dos esforços de teste e identificação das funcionalidades críticas[14].

Ocorre que, as abordagens e ferramentas de teste baseado em risco, apresentam lacunas. Elas trabalham apenas com os itens de riscos, que são representados pelos requisitos de softwares, não levando em consideração os riscos técnicos voltados ao produto de software. Além disso, estão focadas na diminuição do esforço de teste, mais do que com a qualidade específica do sistema. Outro ponto a se destacar, é que não existe uma forma sistemática de definição de qual estratégia de testes deve ser realizada, para reduzir os riscos e impactos voltados à qualidade de software.

Outro ponto a se destacar, é que dentre as abordagens e ferramenta identificadas na literatura [20, 14, 21, 22, 23, 24, 25, 26, 27], existe apenas uma ferramenta [28] que auxilie

na realização da gestão de risco voltada ao teste de software de forma automatizada, entretanto a mesma se enquadra nas lacunas apresentados no parágrafo anterior.

Desta forma, o projeto apresentado, pretende-se responder às seguintes perguntas:

- Como gerir risco técnicos de produtos, nas fases de desenvolvimento de software com foco na qualidade do produto?
- Como focar os esforços de teste e qualidade, nas funcionalidades mais críticas?
- Qual estratégia de teste a ser realizada para garantir a entrega de um software com qualidade ?

Assim sendo, fica evidenciada a necessidade de aprofundamento dos estudos na gestão de risco na qualidade de software a ser aplicada nas fases de desenvolvimento.

## 1.3 Justificativa

A fim de melhorar a qualidade dos sistemas produzidos, faz-se necessário o aprofundamento nos estudos relacionados às técnicas de gestão de riscos em desenvolvimento de software que possibilitem a identificação, análise, avaliação e monitoramento dos riscos do produto de software, com objetivo de minimizar os riscos negativos e aumentar a qualidade do produto e consequentemente a satisfação dos usuários.

A atividade de teste e qualidade de software tem se mostrado fundamental no processo de desenvolvimento, buscando evitar, principalmente, que falhas cheguem aos clientes, deixando-os insatisfeitos e causando prejuízos.

Neste sentido, são antigos os informes referentes a erros de software. Em matéria publicada pela revista *Wired*, são relatados os piores erros da história, alguns deles detalhados[29]:

- Sonda Espacial *Mariner* I: Um *bug* no software de voo do *Mariner* 1 em 1962, fez com que o foguete se desviasse do caminho pretendido no lançamento e o foguete teve que ser destruído sobre o Oceano Atlântico. Tratou-se de um erro nos cálculos da trajetória do foguete.
- Instituto Nacional do Câncer: Em 2000 na Cidade do Panamá, em uma série de acidentes, o software de planejamento terapêutico criado pela *Multidata Systems International*, uma empresa norte-americana, calculava incorretamente a dose adequada de radiação para pacientes submetidos à radioterapia. Pelo menos oito pacientes morreram, enquanto outros 20 receberam overdoses com probabilidade de causar problemas de saúde significativos. Os médicos, que eram legalmente obrigados a checar os cálculos do computador manualmente, foram indiciados por homicídio.

- Gasoduto soviético: Em 1982, a maior explosão registrada na Terra por causas não nucleares teve sua origem numa falha de programação. Supostamente, agentes da Central Intelligence Agency (CIA) infiltraram um *bug* num sistema de informática Canadense adquirido pelos soviéticos para controlar o gasoduto Transiberiano. Seguiam ordens de *Reagan*, que tinha mandado seus agentes sabotar toda a tecnologia Russa, colocando artefatos que permitissem manipular à distância todo tipo de máquinas e tecnologia. Assim, a CIA decidiu sabotar este gasoduto, mas ao ativar o *bug* os resultados obtidos saíram muito diferente do esperado provocando a gigantesca explosão.

Muitos anos se passaram, e as falhas continuam ocorrendo. De acordo com matérias recentemente publicadas, houve um aumento acentuado no número e na gravidade de falhas de software em grandes empresas [30, 31, 32, 33, 15, 34]:

- *Fiat Chrysler - recall* com risco de vida: Em maio de 2017, a Fiat retirou mais de um milhão de caminhões das ruas, devido a uma falha no software que estava ligada a pelo menos um acidente mortal. O problema foi causado por um código errôneo que temporariamente desativou os *airbags* e a funcionalidade do cinto de segurança.

- *Boeing* reconhece pela primeira vez defeitos no software do simulador de voo do 737 MAX: A fabricante de aviões norte-americana *Boeing* admitiu, em matéria recentemente publicada pela Globo, que teve de corrigir falhas no software dos simuladores de voo destinados a formar os pilotos do 737 MAX, o modelo de aeronave envolvido em duas tragédias que deixaram mais de 300 mortos. Segundo a empresa, o software usado nos simuladores era incapaz de reproduzir algumas condições de voo - em especial, aquelas que levaram ao acidente do 737 MAX da Ethiopian Airlines, apenas alguns minutos depois da decolagem. Foram 157 mortos.

- Vários aeroportos - *Check-in* caos: Em setembro de 2017, o caos do aeroporto atingiu o mundo todo quando uma falha de rede global causou uma interrupção no sistema usado para o *check-in* de passageiros e bagagens. Afetou sete aeroportos em sete países diferentes.

- Contas zeradas, Banco de Brasília (BRB): Em maio de 2018, os clientes tiveram suas contas zeradas, devido a uma falha no sistema que zerou o saldo disponível nas contas-correntes e em carteiras de investimentos de acionistas. A falha no sistema provocou pânico nos cliente e filas, e por isso, as agências tiveram de funcionar em horário estendido.

- *British Airways* - Rompimento para 75.000 passageiros: Pela sexta vez no ano de 2017 uma grande falha no software de TI levou a cancelamentos em voos locais e

atrasos significativos em internacionais. Foram necessários três dias de cancelamento do caos para resolver os problemas.

Desta forma, a busca pela qualidade de software é um fator cuja criticidade tem crescido, conforme tecnologias se tornam mais complexas e aumenta a competitividade entre as empresas, e deve ser realizada, uma vez que o domínio de entradas e saídas são diversos, bem como são muitas as possibilidades de caminhos a serem testados e analisados dentro do contexto da qualidade do produto.

Outro ponto a se destacar é que infelizmente o processo de teste e qualidade de software ainda é encarado como um processo que tem início no final da fase de desenvolvimento, desconsiderando recurso e tempo suficiente para o tratamento adequado da qualidade do produto de software. [18].

Com relação aos riscos de software, é bastante comum projetos que enfrentam diversos problemas apresentados por riscos inesperados, não planejados ou simplesmente ignorados. Não obstante, à medida que aumenta a quantidade e complexidade dos sistemas, crescem na mesma proporção a necessidade de utilização de ferramentas e técnicas que permitam uma gestão de risco com vistas a minimizar os impactos de má qualidade do produto.

No cenário atual, são utilizados algumas técnicas de teste baseadas em análise de riscos, que surgiram com a necessidade de minimizar alguns dos problemas relacionados ao esforço da atividade de teste e da gestão de risco, conhecida como *Risk Based Test (RBT)*. Esta abordagem permite a priorização dos esforços e alocação dos recursos para os elementos de software que necessitam ser testados cuidadosamente a partir da identificação, análise e controle dos riscos associados aos requisitos necessários para testar um produto com objetivo de melhorar a sua qualidade.

Ocorre que, mesmo com essas técnicas, encontra-se dificuldade em aplicá-la na prática [18], visto que a análise de risco é entendida como complexa e necessita de conhecimento sobre o produto [35] para sua aplicação.

Outro ponto relevante, é que as abordagens focam bastante na diminuição dos esforços de teste, mas acaba deixando o foco da qualidade do software à desejar. Temos ainda, que a abordagem *Risk Based Test (RBT)* foi desenvolvida para aplicabilidade apenas na fase de teste, que na prática acaba ocorrendo somente ao final do ciclo de vida do desenvolvimento[22] e por muitas vezes os riscos inerentes ao produtos não são gerenciados e consequentemente a qualidade do desejada não é alcançada.

empresa especialista em Teste e Qualidade de software com mais de 23 anos de experiência neste segmento, subsidiou a pesquisa, provendo conhecimento de seus especialistas e na disponibilização de gestão de risco utilizadas nas fases de teste de software.

A RSI Informática (RSI) empresa, especialista em Teste e Qualidade de software e possui mais de 23 anos de experiência neste segmento, subsidiou a pesquisa provendo

conhecimento de seus especialistas e disponibilizando cases de alguns dos seus clientes que aplicaram a gestão de risco voltadas a estas disciplinas.

As ferramentas de gestão de risco com foco no teste e na qualidade do software, não identificam a estratégia de testes a ser realizada. Desta forma, com base nos riscos identificados, quais tipos de testes devem ser realizados?

Essa pesquisa visa contribuir para o preenchimento dessa lacuna, por meio do desenvolvimento de uma ferramenta que permita a identificação, análise, tratamento e monitoramento dos riscos dos produtos de software no intuito de propiciar a melhoria da qualidade dos produtos de software.

## 1.4 Objetivo

O objetivo deste trabalho é desenvolver uma ferramenta de gestão de risco voltada para teste, que possa ser aplicada nas fases de desenvolvimento, com vistas a minimizar os riscos de defeitos e assim contribuir com o aumento da qualidade dos produtos de softwares.

### 1.4.1 Objetivos Específicos

O objetivo geral foi dividido em objetivos específicos, que considerados em seu conjunto contribuirão para o alcance do resultado final da pesquisa. São eles:

- Identificar os pontos positivos e negativos das abordagens atuais, voltadas a gestão de risco no teste e qualidade de software, utilizadas nas fases de desenvolvimento para embasamento dos requisitos da ferramenta;
- Construir uma ferramenta automatizada que gerencie os riscos técnicos do produto no ciclo de desenvolvimento de software;
- Validar a aplicabilidade da ferramenta por meio de estudos de caso.

Essa pesquisa é de caráter exploratório com a utilização de métodos, técnicas e outros procedimentos científicos. Foi desenvolvida com o apoio da revisão da literatura, *frameworks*, leis e normas objetivando apresentar um resultado embasado na literatura a partir de modelos de sucessos já definidos na indústria de TI. Para tanto, a pesquisa está estruturada da seguinte forma: No Capítulo 2 apresenta-se uma revisão detalhada da literatura, contendo conceitos de Engenharia de software, Qualidade e teste de software e Gestão de Riscos. No Capítulo 3 é exibe-se a metodologia de pesquisa a ser utilizada neste estudo, no Capítulo 4 ressalta-se a ferramenta proposta nesta pesquisa, no Capítulo 5 expõem-se os estudos de caso da pesquisa, no Capítulo 6 demonstram-se os resultados

da aplicação da ferramenta e por fim, no Capítulo 7 elenca-se a conclusão e proposições de trabalhos futuros.

# Capítulo 2

# Revisão da Literatura

Esse capítulo apresenta uma visão geral acerca do tema deste trabalho e expõe os conceitos fundamentais, técnicas e ferramentas que servirão como base para o desenvolvimento desta pesquisa.

A revisão de literatura está estruturada em 6 seções, sendo que a primeira apresenta os conceitos relacionados à engenharia de software e seus modelos de desenvolvimento; a segunda, sobre qualidade de software; a terceira, sobre testes de software e seus processos; a quarta, sobre os conceitos de gestão de riscos e ainda os riscos na engenharia de software voltadas ao produto; e, por fim, sobre ferramentas de testes de software baseados em risco que foram base para o desenvolvimento da ferramenta.

## 2.1 Engenharia de software

O termo Engenharia de software, foi criado na década de 1960, e utilizado oficialmente em 1968 na *NATO Science Comittee.*[36] Na oportunidade, tinham-se interesse por métodos padronizados a serem usados no local de trabalho com o intuito de criar aplicações de qualidade. Neste ponto, já haviam se tornado um interesse importante para muitas organizações.

Segundo Pressmam,[37, p.17] a Engenharia de software é a criação e a utilização de sólidos princípios de engenharia a fim de obter software econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais. É uma área de conhecimento composta por teorias, métodos e conjuntos de ferramentas necessários à geração de um produto de software[38].

O desenho de soluções de software, para dar resposta à crescente complexidade e diversidade dos sistemas, bem como às alterações nos requisitos [39] durante o seu ciclo de desenvolvimento, necessita do auxílio de uma área de conhecimento que melhore as garantias oferecidas quanto à qualidade do produto, sendo ela a engenharia de software.

São vários os modelos do processo de desenvolvimento de um software. O processo pode ser diferente dependendo da organização ou do projeto, ou seja, ele é adaptável às necessidades [40]. Esse processo conta com a apoio de toda a equipe de desenvolvimento, equipe de testes, gerentes, entre outros, além dos próprios solicitantes do software que devem colaborar com a definição, construção e teste.

Não existe modelo de desenvolvimento de software mais ou menos corretos, mas sim processos de desenvolvimento mais ou menos adequados à complexidade do projeto em questão, ao tipo de equipe que o pratica, e ao tipo de utilização pretendido para o produto de software resultante [41]. Quando se seleciona um processo de desenvolvimento inadequado, os resultados obtidos, ou seja, o produto final de software, tenderão a possuir uma qualidade inferior do que se se usasse um modelo de processo adequado.

Assim sendo, serão abordados nas seções 2.1.1 a 2.1.5 os principais modelos e metodologias utilizadas no ciclo de desenvolvimento que podem ser utilizados na construção e desenvolvimento de um software. A ferramenta proposta nesta pesquisa, será aplicada a qualquer modelo de desenvolvimento de software, com vistas a reduzir a probabilidade e impacto da entrega de um produto com baixa qualidade.

### 2.1.1 Modelo de Desenvolvimento em Cascata

O modelo de desenvolvimento em cascata, também conhecido como sequencial, é um modelo clássico da engenharia de software. É um dos modelos mais antigos e ainda é utilizado em muitos projetos, incluindo no governo e em muitas empresas citemunassar2010comparison.

Este modelo é utilizado principalmente quando os requisitos de um determinado problema são bem compreendidos no início do desenvolvimento. Também pode-se utilizar o modelo cascata quando um software necessita de uma nova funcionalidade e os requisitos estão bem definidos e são estáveis.

O ciclo de vida cascata consiste em vários estágios sequenciais, conforme apresentado na Figura 2.1. Dessa forma, o modelo começa com o levantamento de requisitos e necessidades junto às partes interessadas; depois vamos para a fase de planejamento onde definimos estimativas, cronograma e acompanhamento; então partimos para a modelagem onde fazemos a análise e projeto; seguindo para a construção, na qual codificamos; logo após, testamos para garantir a qualidade; em seguida passamos para a implantação, entrega e suporte do software concluído e emprego de melhorias.

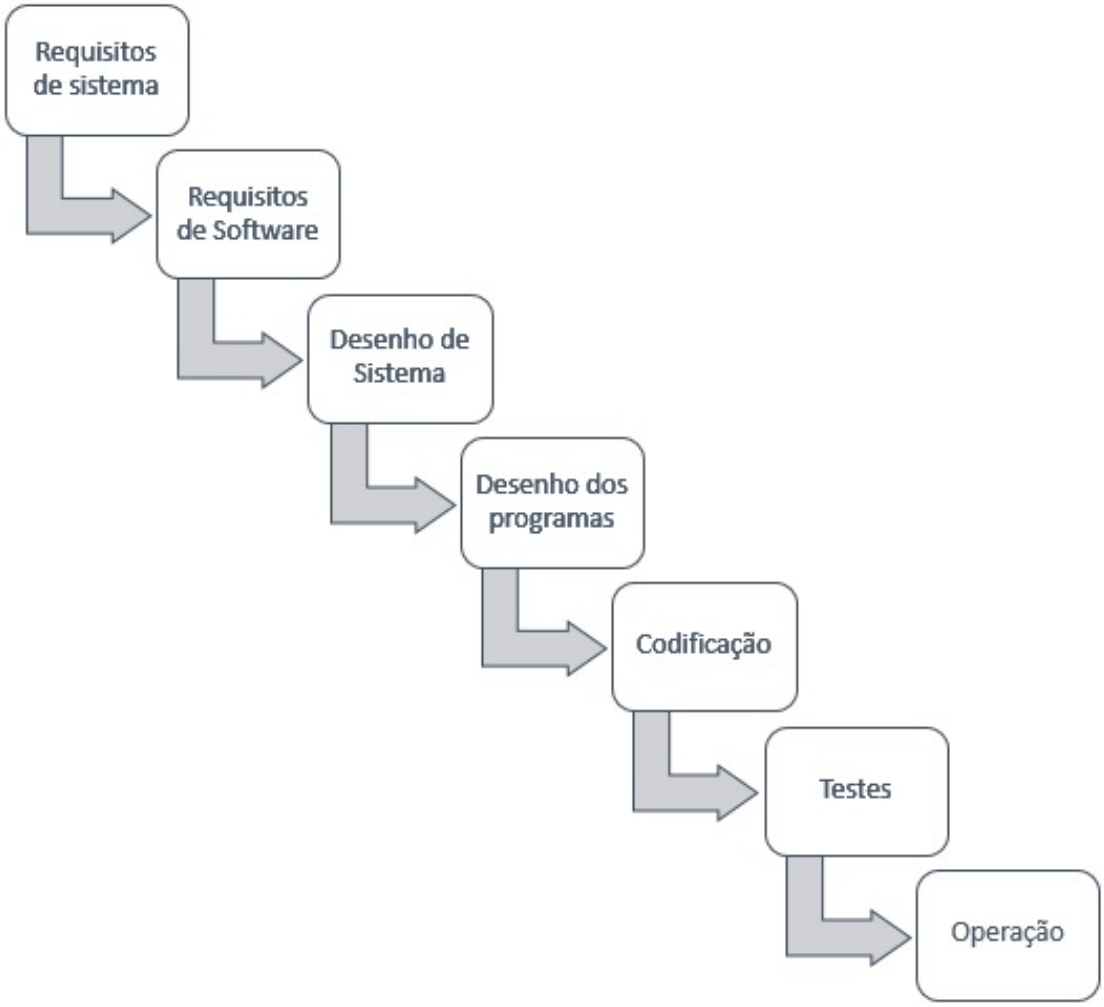



Figura 2.1: Modelo de desenvolvimento de software em cascata
Fonte: Adaptado [42]

Como se pode verificar na Figura 2.1, o modelo apresenta as seguintes atividades descritas resumidamente [42]:

- **Requisitos de sistema:** Estabelece os componentes para construir o sistema, incluindo os hardwares, requisitos, ferramentas de software e outros componentes.
- **Requisitos de software:** Estabelece os requisitos para as funcionalidades do software. Inclui a determinação das integrações, base de dados, desempenho, requisitos de interface do usuário entre outros.
- **Desenho do sistema:** Determina a estrutura do software para atender às necessidades dos requisitos. Define os principais componentes e interação desses.

- **Desenho do Programa:** Analisa os componentes de software definido no estágio desenho do sistema e produz a arquitetura do software de acordo com cada componente a ser implementado.
- **Codificação**: Implementa o design detalhado e especificações de acordo com os requisitos definidos.
- **Teste:** Determina se o software atende aos requisitos especificados e tem como objetivo identificar quaisquer erros presentes no código.
- **Manutenção:** Resolve problemas e melhorias solicitadas após o lançamento do software.

O modelo cascata não proíbe o retorno a um fase anterior, entretanto, isso envolve retrabalho e alto custo. Cada fase concluída requer revisão formal e extenso desenvolvimento de documentação. Dessa forma, equívocos realizados na fase de requisitos tornam-se caros para corrigir nas fases posteriores do projeto.

Como dito, é um modelo clássico, que tem servido como base para muitos outros modelos de ciclo de vida de desenvolvimento de software, inclusive ao em V, apresentado na seção 2.1.2.

### 2.1.2 Modelo de Desenvolvimento em V

Foi apresentado a primeira vez no Congresso NCOSE em 1991 em *Chattanooga* e foi desenvolvido pela National Aeronautics and Space Administration (NASA). Assim como o modelo em cascata, o ciclo de vida em forma V, segue um caminho sequencial de execução de processos[42].

Este modelo é seguido de maneira *top-down*e faz uma relação entre ações da garantia da qualidade atividades de construção de software [40], conforme apresentado na Figura 2.2. O envolvimento do teste é incluído no início do ciclo de vida antes que qualquer codificação seja feita, durante cada fase que precede a implementação. [42]

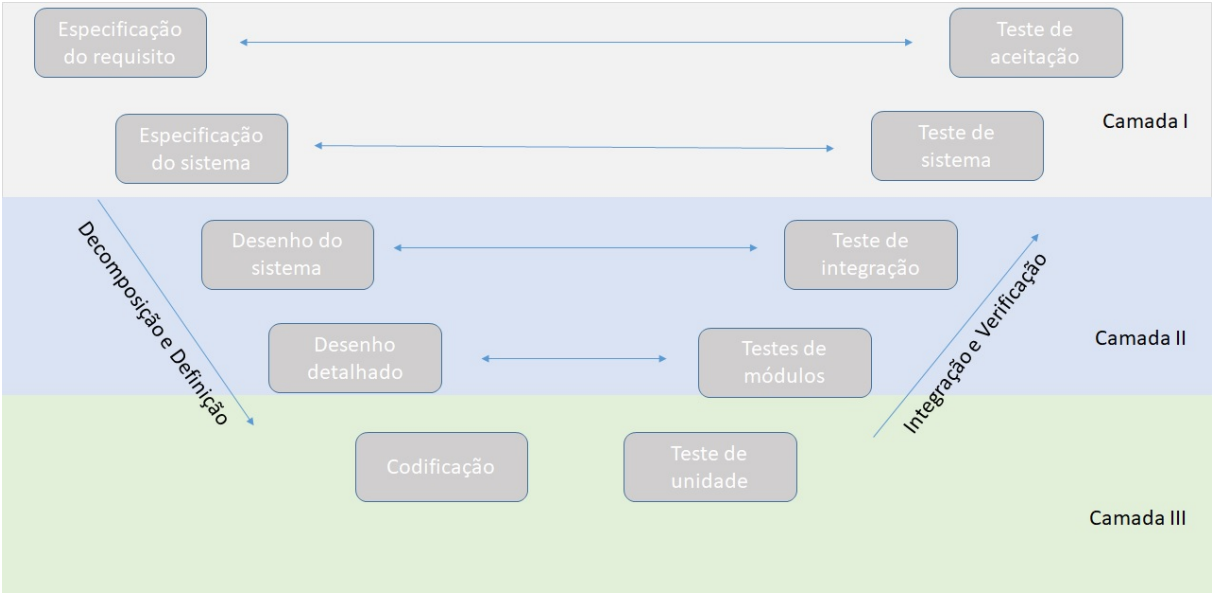



Figura 2.2: Modelo V
Fonte: Adaptado [40]

Na camada I, é realizado a descrição das funções do sistema e dos requisitos referente às funcionalidades, através das atividades de "Especificações de requisitos"e "Especificações de sistemas". Através das atividades de "Teste de aceitação"e "Teste de sistema"é verificado se o sistema entregue está de acordo com especificação de sistema e requisitos.

Na camada II, é realizada a arquitetura do software e descreve a estrutura de infra e funcional. Por meio do desenho do sistema, são identificamos os principais componentes, relações lógicas e estrutura de dados arquitetural. Neste ponto são realizados os testes de integração e testes nos módulos, que têm por objetivo encontrar falhas de integração entre as unidades.

A camada III é a implementação da arquitetura e escrita do código fonte, através dos requisitos funcionais e não funcionais. Neste ponto são realizados testes de unidade, em que é testada cada unidade de maneira isolada, no próprio código fonte.

Esse modelo é de simples utilização, cada fase tem seu resultado específico e funciona bem, assim como modelo tradicional para projetos onde os requisitos são bem definidos. Entretanto é rígido como o modelo cascata, ao qual existe pouca flexibilidade a mudanças no escopo, que acaba sendo cara, devido ao seu processo [42].Com isso, o modelo espiral foi desenvolvido de modo a combinar as melhores características dos modelos Linear e de prototipação, e será apresentado na seção 2.1.3.

### 2.1.3 Modelo de Desenvolvimento em espiral

O modelo espiral foi definido por Barry Boehm [43] na tentativa de modificar o modelo em cascata, introduzindo pequenas iterações.

A mitigação de riscos é algo muito importante para o modelo em espiral e isso é feito por meio da construção de protótipos e obtenção do *feedback* dos usuários. O modelo espiral foi o primeiro a reconhecer a necessidade de se gerenciar os riscos do projeto de forma explícita.[42]

O modelo em espiral representa uma mudança de paradigma da abordagem baseada em especificação para uma abordagem baseada em risco. Cada ciclo atravessa quatro quadrantes,apresentado na Figura 2.3, sendo eles:

- Determinação dos objetivos.
- Avaliação e redução de riscos.
- Desenvolvimento e validação.
- Planejamento da próxima iteração.

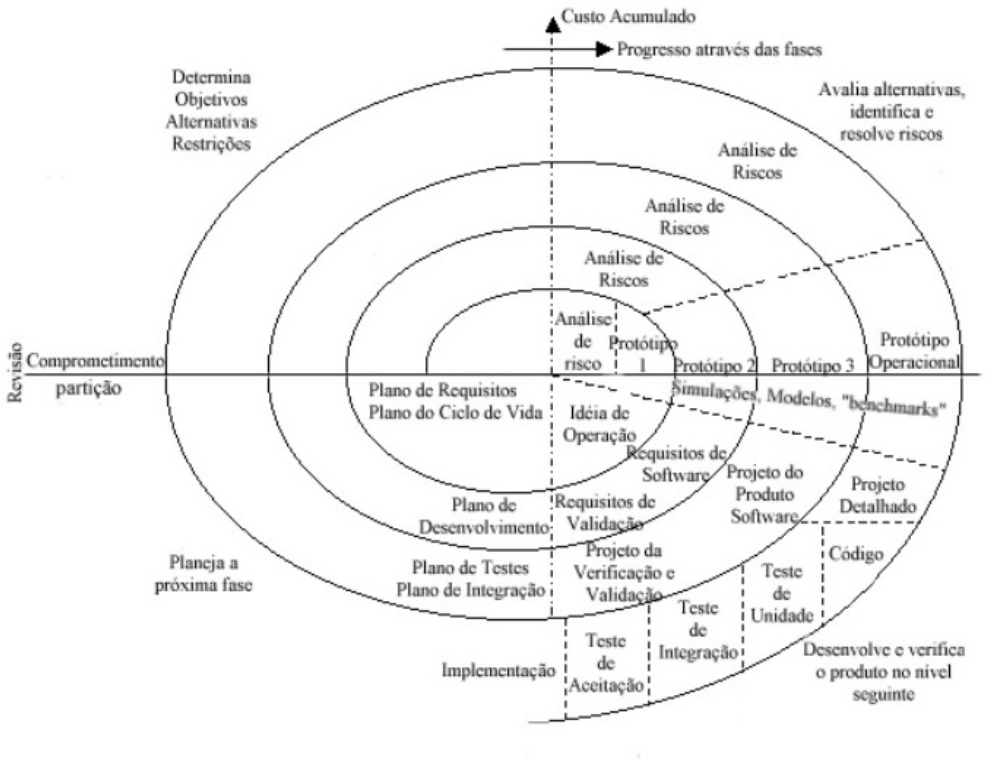



Figura 2.3: Modelo Espiral
Fonte: Adaptado [42]

Conforme cada ciclo dentro do espiral evolui, o protótipo é construído, verificado em relação aos requisitos e validado através de testes. A gestão de risco é utilizada para determinar a quantidade de tempo e esforço a ser gasto nas atividades, durante o ciclo de vida, sendo eles: planejamento, gerenciamento de projetos, gerenciamento de configuração, garantia de qualidade, verificação formal e testes. Dessa forma, o gerenciamento de risco é utilizado como uma ferramenta de controle de custo a cada ciclo de vida [40].

Este modelo permite uma gestão de risco em seu ciclo de desenvolvimento e se mostra eficiente para projetos grandes e críticos e ainda permite o desenvolvimento do software desde o início do seu ciclo de vida. Entretanto, se torna caro fazer gestão de risco, pois requer conhecimentos altamente especializados e o sucesso do projeto depende da fase de gestão de risco o que o torna ineficiente para projetos de menores portes [44].

Esse foi o primeiro a abordar o modelo incremental, porém com aspectos sistemáticos e controlados do modelo cascata. Será apresentado o modelo de desenvolvimento iterativo e incremental na seção 2.1.4.

### 2.1.4 Modelo de Desenvolvimento Iterativo e Incremental

O Desenvolvimento Iterativo e Incremental é um dos clássicos modelos de processo de desenvolvimento de software criado em resposta às fraquezas do modelo em cascata [42].

Este modelo tem por finalidade a entrega de funcionalidades parciais, onde cada versão do produto entregue é semelhante ao anterior, mas integrando novas funcionalidades. Desta forma é possível já ir obtendo *feedback* dos usuários do sistema.

Este modelo combina, elementos do modelo em cascata de uma forma iterativa. Além disso, cada sequência linear produz incrementos entregáveis do software[45]. É um método estagiado, ao qual em várias partes do sistemas, têm um desenvolvimento em paralelo e integrado quando completas, conforme apresenta a Figura 2.4.

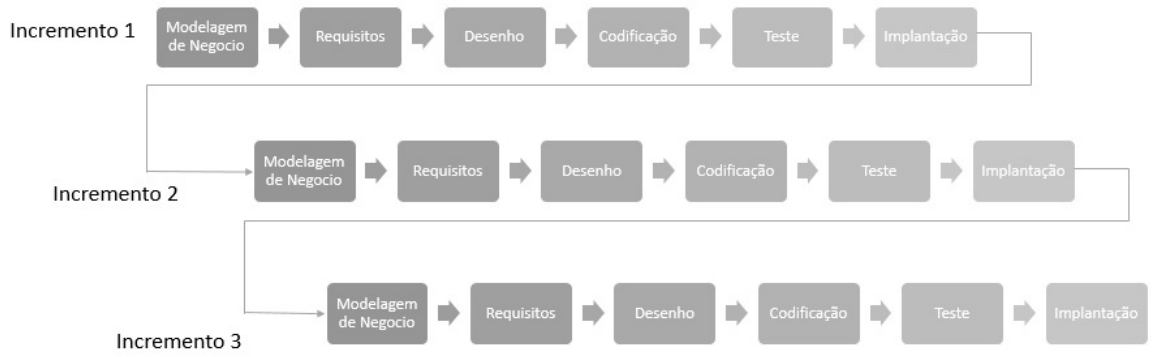



Figura 2.4: Modelo Iterativo Incremental

Os dois padrões mais conhecidos de sistemas iterativos de desenvolvimento são o RUP (Processo Unificado da *Rational*) e o Desenvolvimento ágil de software, que serão apresentados tópicos seguintes.

### Modelo de Processo Unificado

O processo Unificado da *Rational* conhecido como Rational Unified Process (RUP), é um *framework* da engenharia de software criado para apoiar o desenvolvimento orientado a objetos, fornecendo uma forma sistemática para se obter vantagens no uso da Unified Modeling Language (UML) [46]. Foi criado pela *Rational Software Corporation* na década de 90 e adquirido em fevereiro de 2003 pela International Business Machines (IBM).

Este surgiu com as lacunas apresentadas pelo modelo cascata, com a capacidade de lidar eficazmente com problemas comuns em projetos de desenvolvimento de software[43, 47]. Essas limitações, incluíam ausência de gestão de risco, baixa reutilização, ausência de controle de versões entre outros.

Este *framework* é fundamentado no conjunto de melhores práticas, sendo elas: desenvolver software de forma iterativa, gerenciar os requisitos, utilizar a arquitetura baseada em componentes, modelar o software de forma visual,verificar a qualidade do software e controlar as mudanças de software[40].

O modelo apresenta as seguintes perspectivas[48]:

- Perspectiva dinâmica, que apresenta as fases do RUP ao longo do tempo. Os processos mostrados nesta perspectiva são dinâmicos, ou seja, são mudando constantemente.
- Perspectiva estática, mostra os aspectos estáticos das fases do RUP.
- Perspectiva prática, é feita das boas práticas utilizadas durante o processo.

A Figura 2.5, apresenta o modelo RUP em suas dimensões:

1. O eixo horizontal reflete o tempo e apresenta os aspectos dinâmicos do processo. Representa os ciclos, fases, iterações e marcos.
2. O eixo vertical reflete os aspectos estáticos do processo e representa as atividades, artefatos, trabalhadores e fluxos de trabalho.

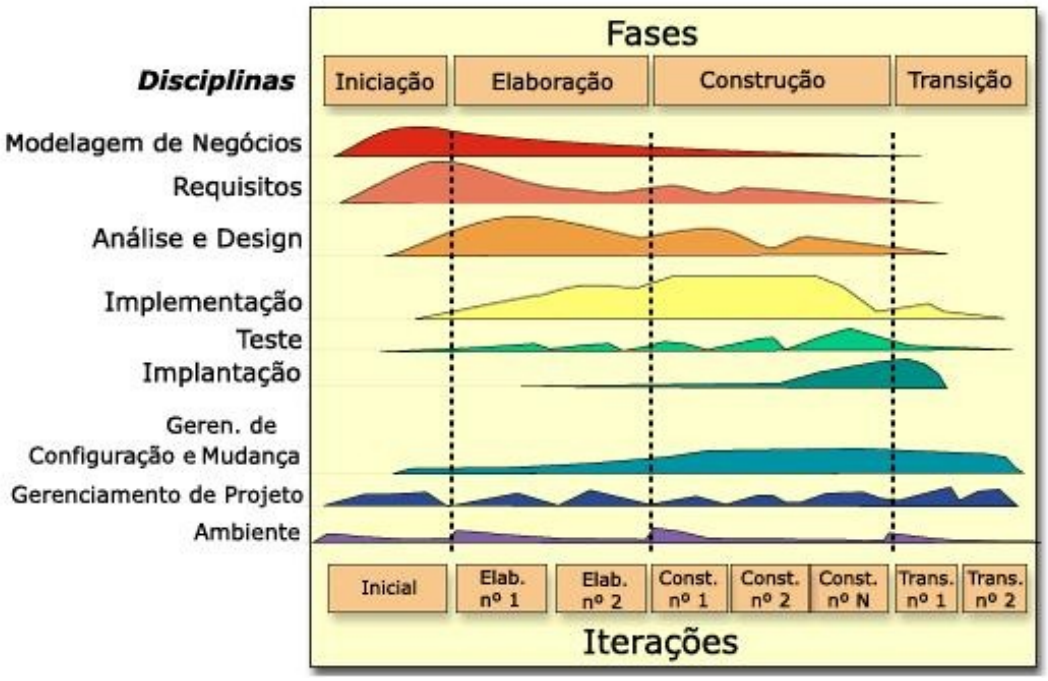



Figura 2.5: RUP - Processo Unificado
Fonte: [49]

O modelo apresentado pela Figura 2.5 possui quatro fases, sendo elas: Iniciação, Elaboração, Construção e Transição, ao qual cada fase tem mais de uma iteração e é concluída com marcos.

O RUP usa modelos extensos e são embasamento na modelagem visual através da UML, que compreende em notações gráficas semi-formais que se tornou um padrão mundial para modelagem orientada a objetos [50]. Essas modelagens facilitam a abstração e a construção de várias do software e suporta modelagem estática e dinâmica. As notações incluem diagramas de caso de uso, atividade, classe, objeto, interação, estado entre outros.

Em muitos casos não é possível obter todos os requisitos no início do projeto. É muito difícil definir todo o problema completamente, criar uma solução, criar o software e testá-lo em ordem sequencial. Por este motivo, este modelo permite que software seja desenvolvido iterativamente, ou seja, você pode voltar quantas vezes precisar para alterar e atualizar o software. A necessidade de desenvolvimento iterativo surge naturalmente à medida que você ganha uma compreensão crescente do problema. O RUP aborda os itens de maior risco em cada estágio do ciclo de vida. Isso ajuda a diminuir o risco geral do projeto. [48].

Desta forma, segundo Hirsch e Hanssen[51, 52], o modelo RUP é bem robusto no que tange tange funções, fluxos e artefatos, no entanto, apesar de otimizável torna-se complexo, para projetos pequenos, devido a várias documentações que precisam ser pro-

duzidas e o modelos que precisam ser aplicados. Desta forma, surge o manifesto ágil, apresentando modelos iterativos e incrementais menos robustos, conforme apresentado na seção 2.1.5.

### 2.1.5 Ágil

O desenvolvimento ágil surgiu de uma constatação realizada de forma independente por diversos profissionais renomados na área de engenharia de software. Estes profissionais, a maioria veteranos consultores em projetos de software, decidiram reunir-se no início de 2001 durante um *Workshop* realizado em *Snowbird, Utah, EUA*. Este encontro foi um marco denominado de Manifesto Ágil de "Desenvolvimento de software"[53].

Este manifesto estabelece que o desenvolvimento ágil deve ter foco em quatro valores fundamentais:

- 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.

O desenvolvimento de software utilizando metodologias ágeis apoia na produção de um sistema que responda a volatilidade das mudanças. Essas fornecem oportunidades para avaliar o impacto da inclusão de mudanças dos requisitos durante o desenvolvimento de software em todo seu ciclo de vida. A construção ocorre em intervalos curtos e lançamentos de software são feitos para capturar pequenas mudanças incrementais [40].

#### Extreme Programming (XP)

É uma metodologia que enfatiza a velocidade e simplicidade no desenvolvimento de software. É fundada nos valores apresentados na Figura 2.6 [54]:

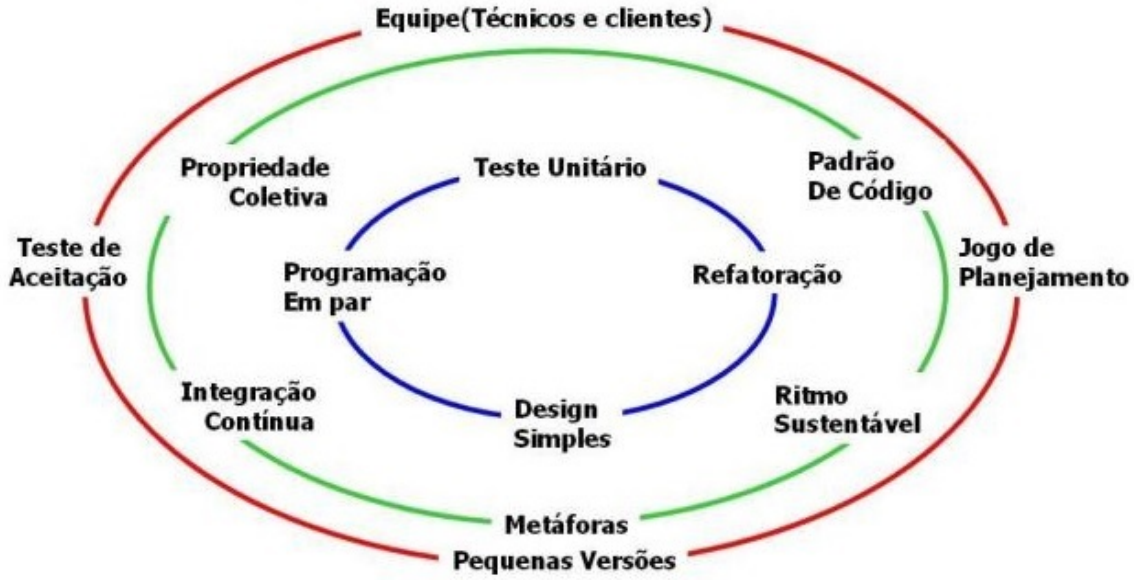



Figura 2.6: Etapas da metodologia XP - *Extreming Programing*
Fonte: [55]

As etapas da metodologia XP, apresentada na Figura2.6 contemplam:

**Jogo do Planejamento:** Os clientes decidem o escopo e prazo de lançamentos baseados nas estimativas fornecidas pelos desenvolvedores. Neste sentido os desenvolvedores implementam apenas as funcionalidades exigidas pelas histórias nesta iteração.

**Pequenos lançamentos:** O software é lançado em produção em poucos meses. Novas liberações são feitas frequentemente com incrementos.

**Metáfora:** A forma do sistema é definida por uma metáfora ou conjunto de metáforas compartilhadas entre o cliente e desenvolvedores.

**Simplicidade de Projeto:** O código está, a qualquer momento, na forma mais simples e mais clara, conforme os padrões definidos pela equipe de desenvolvimento. Não contém código duplicado e tem menos classes e métodos possíveis. Esta regra pode ser resumida como "Diga tudo de uma vez só uma vez."

**Testes:** Os desenvolvedores escrevem os testes unitários que são definidos antes da codificação, os testes dos métodos críticos do software ou métodos simples que podem apresentar alguma exceção de processamento. Os clientes escrevem testes funcionais através dos critérios de aceite e as histórias em uma iteração. Esses testes também devem ser executados, embora a decisão deve ser feita comparando o custo de envio de um defeito conhecido e o custo do atraso.

**Refatoração**: A cada nova funcionalidade adicionada, é trabalhado o design do código até ficar na sua forma mais simples, mesmo em código já existente e entregue.

**Programação em par:** Toda a produção de código é escrita por duas pessoas em único monitor/teclado/mouse, somando forças para a implementação do código.

**Propriedade coletiva:** Qualquer desenvolvedor melhora qualquer código em qualquer lugar a qualquer momento, se eles enxergarem a oportunidade. O código é de toda equipe!

**Cliente presente:** Constante disponibilidade do cliente para colaborar em dúvidas, alterações, e prioridades em um escopo, ou seja, dando um dinamismo ativo ao projeto.

**Semana de 40 horas:** Ninguém pode trabalhar uma segunda semana consecutiva de horas extras. Até mesmo horas extras isoladas usadas com muita frequência é um sinal de mais profundos problemas que devem ser abordados.

**Rodízio de pessoas:** Duplas de desenvolvedores são revezadas periodicamente, com o objetivo de uniformizar os códigos produzidos, deixar todos os módulos do sistema com mesmo padrão de código e pensamento e compartilhar o código com todos da equipe.

**Regras:** Todos os participantes da equipe precisam seguir a regra, que pode mudar a qualquer momento, desde que todos concordem e avaliem o impacto das mudanças.

Essas boas práticas citadas não são estáticas, o XP permite liberdade, dando dinamismo na forma de trabalho, conforme necessidade exigida. É muito comum que essas práticas sejam alinhadas a outras e com experiência de projetos anteriores oriundos de outras metodologias, tais como SCRUM, RUP e outros.

Nessa seção foram apresentadas as principais características dos modelos de desenvolvimento de software, sendo percebido que com a utilização de um modelo de desenvolvimento é possível aumentar a qualidade e produtividade, permitindo um maior controle sobre este processo. A ferramenta proposta nesta pesquisa, poderá ser utilizada em qualquer dos modelos apresentados neste capítulo. A seção 2.2, irá abordar a qualidade de software.

## 2.2 Qualidade de software

Devido a expansão dos sistemas dependentes de software, tem aumentado cada vez mais a necessidade de abordagens destinadas a garantir e verificar a qualidade dos produtos de software [56]. A qualidade de software é uma área de conhecimento da Engenharia que tem como objetivo a garantia da qualidade através da definição e normalização de processos de desenvolvimento.

Existem muitas definições de qualidade de software dispostas na literatura, sob diferentes pontos de vista de vários autores.

Segundo Chandrupatla[57], a qualidade é percebida de forma diferente por pessoas diferentes. A ideia de qualidade aparentemente é intuitiva; contudo, quando examinado

mais longamente, o conceito se revela complexo. Definir qualidade para estabelecer objetivos é, uma tarefa menos trivial do que aparenta a princípio.

Do ponto de vista de Pressman [37], qualidade de software é o atendimento a conformidade quanto aos requisitos funcionais e de desempenho que foram explicitamente declarados, aos padrões de desenvolvimento claramente documentados, e às características implícitas que são esperadas de todo software desenvolvido por profissionais.

De acordo com Crosby,[58] "a qualidade é a conformidade aos requisitos". Esta definição nos é bastante interessante, pois evidencia qual o caminho a seguir para julgar a qualidade de um software.

Existem várias maneiras de definir qualidade de software, não sendo nenhuma perfeita, pois é um conceito relativo e dinâmico. Entretanto, pode ser medida e comparada com padrões previamente definidos. Quando um item é avaliado em características mensuráveis, é possível identificar os tipos de qualidade. No âmbito da qualidade do produto, é estabelecida quanto ao nível de atendimento às suas especificações [37].

Neste sentido, serão base para esta pesquisa três pontos de qualidade, voltadas ao produto, indicados por Pressman [37]:

- Requisitos de software, são a base para se realizar medidas de qualidade. A falta de conformidade com as especificações é um problema de ausência de qualidade.
- Padrões especificados definem um conjunto de critérios de desenvolvimento que normatizam a forma com que o software será desenvolvido.
- O software deve obedecer requisitos que muitas vezes estão implícitos, mais são esperados de um software. A ausência desses itens tornam a qualidade de software discutível.

A qualidade dificilmente é alcançada através de um produto já pronto. O ponto forte é prevenir o nível de defeitos e deficiências, obtendo qualidade a partir de medições, com estruturação de processos, métodos, ferramentas e técnicas[35]. Nesse sentido, a garantia da qualidade de software se apoia no fornecimento da visibilidade da eficácia do processo que está sendo utilizado pelo projeto de desenvolvimento, bem como da qualidade do produto que está sendo desenvolvido, conforme apresentado na seção 2.2.1.

### 2.2.1 Garantia da Qualidade de Software

Segundo Pressman[37, p.387] a garantia de qualidade de software, consiste em criar um conjunto de atividades que ajudem a garantir que todo artefato resultado da engenharia de software apresente alta qualidade, além de realizar as atividades de garantia e controle

em todos os projetos e usar métricas para desenvolver estratégia para aperfeiçoar a gestão da qualidade.

Dessa forma, o *Software Quality Assurance (SQA)* em sua tradução Garantia de Qualidade de Software, envolve todo o processo de desenvolvimento de software abrangendo o monitoramento e melhorias de processos pertinentes que a garantia dos padrões sejam seguidos e garantindo que problemas sejam encontrados e ações corretivas sejam tomadas. Portanto, consiste em realizar a qualidade tanto do processo quanto do produto.

Um trabalho de garantia de qualidade abrange seis dimensões para métodos e ferramentas de desenvolvimento que são [35]:

- Revisões formais;
- Estratégias de testes;
- Controle de documentação;
- Histórico de manutenções;
- Procedimento de adequação a padrões de desenvolvimento;
- Mecanismos de medição;

Lewis [59] cita as atividades mais comuns do SQA, sendo estas categorizadas como: Teste de software (Verificação e Validação), Gerenciamento de Configuração de software e Controle da Qualidade.

Nesse sentido, a garantia de qualidade de software resume-se em um conjunto de atividades necessárias para proporcionar a confiança de que os processos sejam praticados, melhorados continuamente e organizados de uma maneira que possa atender as especificações do escopo do produto, visando atender as necessidades de uso do software [5].

A garantia de qualidade de software é atingida através de um plano de SQA que estabelece os métodos a serem utilizados no projeto, visando garantir que os artefatos e produtos sejam gerados e revisados a cada passo do projeto [38].

Segundo Vilas Boas[60], a avaliação de produto de software baseada em normas de qualidade tem sido uma das formas empregadas por organizações que produzem software para aferirem a qualidade de seus produtos. Para que a avaliação seja mais efetiva, é importante a utilização de modelos de qualidade que permitam estabelecer e avaliar requisitos de qualidade, e que o processo de avaliação seja bem definido e estruturado.

Assim, para que se possa comparar a qualidade de produtos de software, foram criados padrões, os quais são denominados modelos de qualidade. Esses modelos são descritos na seção 2.2.2.

### 2.2.2 Modelos de Qualidade de Software

Os modelos de qualidade são fundamentais para se avaliar a qualidade dos produtos de softwares em diversas situações. São vários os modelos abordados na engenharia de software que abrangem características de qualidade do produto. A seguir é apresentada uma breve descrição dos principais modelos [61, 62].

***McCall's Quality Model***

É um dos modelos mais conhecidos na literatura da engenharia de software. Este modelo apresenta três perspectivas distintas com relação a qualidade com base em requisitos, são elas: Habilidade de ser alterado, Adaptabilidade a Novos Ambientes, Características Operacionais.

***Boehm's Quality Model***

Este modelo tenta definir qualitativamente a qualidade do software por meio de um conjunto predefinido de atributos e métricas. Consiste em três níveis de características, sendo elas: Utilidade, que aborda o quão fácil de usar, confiável e eficiente é o software; Sustentabilidade, que aborda como é fácil entender, modificar e testar novamente o produto; e Portabilidade, que abrange se é possível usar o produto quando o ambiente for alterado [63] .

***Dromey's Quality Model***

Este modelo de qualidade propõe que a qualidade difere para cada produto. É focado na relação entre os atributos de qualidade e os sub atributos do produto com o software. Este modelo é baseado nos atributos de funcionalidade, confiabilidade, manutenibilidade, reusabilidade e portabilidade e é divido em 4 propriedades: concreto, interna, contextual e descritiva, cada uma delas contendo atributos de qualidade [64].

***FURPS Quality Model***

Este modelo indica requisitos como restrições de projeto, requisitos de implementação, requisitos de interface e requisitos físicos e são representados por cinco características, sendo elas: Suportabilidade, Funcionalidade, Usabilidade, Confiabilidade e *Performance*.

***ISO 9126 Quality Model***

Aborda uma estrutura de modelo de qualidade que divide o relacionamento de diferentes perspectivas para qualidade. Define uma abordagem para qualidade no ciclo de vida de desenvolvimento de software que abrange a qualidade no ponto de vista de medidas de processo, medidas internas, medidas externas e medidas de qualidade em uso, conforme apresentado na Figura 2.7 Essas medidas asseguram que através do processo, consiga se assegurar a qualidade do processo a qualidade interna e externa e a qualidade em uso [65] Quando tratamos de qualidade interna e externa, estamos tratando de características dentro do ciclo de desenvolvimento, ou seja, momento em que o software que ele está sendo

construído. Quando falamos de Qualidade em uso, é a visão da qualidade do produto de software do ponto de vista do usuário na utilização do software.

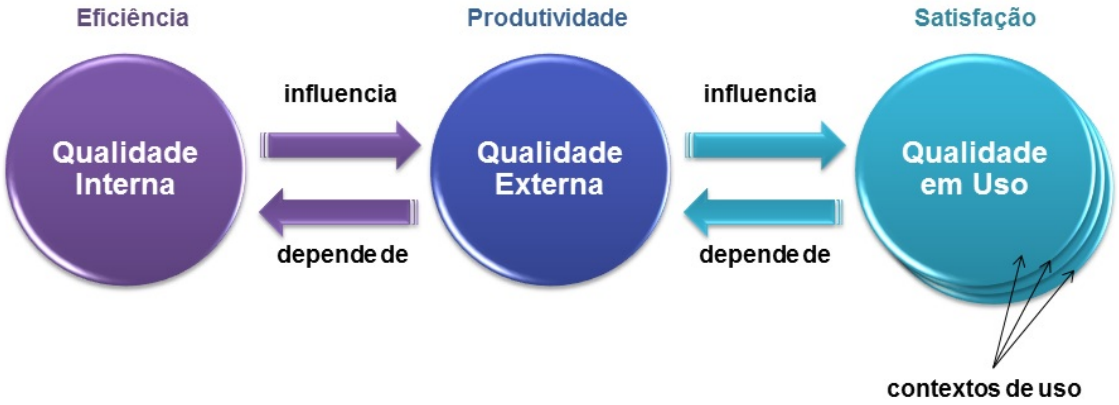



Figura 2.7: Modelo de Qualidade
Fonte: Adaptado [65]

Esta norma foi evoluída para a norma 25010 em 2011, por meio da Square, conforme apresentado na próxima seção.

***Square 25010 - Software Quality Requirements and Evaluation, SQuaRE***

É uma evolução das normas ISO/IEC 9126 [65, 2] e aborda as mesmas dimensões de qualidade, sendo elas a qualidade interna, externa e qualidade em uso, entretanto com uma visão mais abrangente.

A qualidade é o grau em que o sistema satisfaz as necessidades declaradas e implícitas de suas várias partes interessadas fornece valor agregado. São representados no modelo de qualidade através da categorização da qualidade do produto através de características e subcaracterísticas. A Figura 2.8 representa as características de qualidade interna e externa.


Figura 2.8: Características de Qualidade Interna e Externa - ISO25010
Fonte: Adaptado [2]

Este modelo, determina quais características de qualidade serão levadas em consideração ao avaliar as propriedades de um produto de software. O modelo de qualidade do produto definido na ISO / IEC 25010 compreende as oito características de qualidade que foram apresentadas na Figura 2.8 e estão listadas a seguir [2]:

### Adequação Funcional

Esta característica é representada pelo grau em que um produto ou sistema fornece suas funções que atendem às necessidades declaradas e implícitas quando utilizadas sob condições especificadas. Essa característica é composta das seguintes sub-características [2]:

- **Completude funcional:** Grau para o qual o conjunto de funções abrange todas as tarefas especificadas e objetivos do usuário.
- **Correção funcional:** Grau para o qual um produto ou sistema fornece os resultados corretos com o grau de precisão necessário.
- **Adequação funcional:** Grau para o qual as funções facilitam a realização de tarefas e objetivos específicos.

### Eficiência de desempenho

Esta característica é representada pelo grau do desempenho em relação à quantidade de recursos usados nas condições declaradas. Essa característica é composta das seguintes subcaracterísticas [2]:

- **Comportamento temporal:** Grau em que os tempos de resposta e processamento e as taxas de rendimento de um produto ou sistema, ao executar suas funções, atendem aos requisitos.
- **Utilização de recursos:** Grau para o qual um produto ou sistema fornece os resultados corretos com o grau de precisão necessário.
- **Capacidade:** Grau para o qual os limites máximos de um produto ou parâmetro do sistema atendem aos requisitos.

### Compatibilidade

Esta característica é representada pelo grau com que um produto, sistema ou componente pode trocar informações com outros produtos, sistemas ou componentes e / ou executar suas funções necessárias, enquanto compartilha o mesmo ambiente de hardware ou software. Essa característica é composta das seguintes subcaracterísticas [2]:

- **Coexistência.** Grau para o qual um produto pode executar suas funções necessárias de maneira eficiente, compartilhando um ambiente e recursos comuns com outros produtos, sem afetar negativamente qualquer outro produto.
- **Interoperabilidade:** Grau para o qual dois ou mais sistemas, produtos ou componentes podem trocar informações e usar as informações que foram trocadas.

### Usabilidade

É representada pelo grau para o qual um produto ou sistema pode ser usado por usuários específicos para atingir metas especificadas com eficácia, eficiência e satisfação em um contexto específico de uso. Essa característica é composta das seguintes subcaracterísticas [2]:

- **Reconhecimento de adequabilidade:** Grau em que os usuários podem reconhecer se um produto ou sistema é adequado às suas necessidades.
- **Aprendizagem:** Grau em que um produto ou sistema pode ser utilizados por usuários específicos para atingir as metas especificadas de aprender a usar o produto ou sistema com eficácia, eficiência, isenção de riscos e satisfação em um contexto específico de uso.

- **Operabilidade:** Grau para o qual um produto ou sistema possui atributos que facilitam a operação e o controle.
- **Proteção contra erros do usuário:** Grau para o qual um sistema protege os usuários contra erros.
- **Estética da interface do usuário:** Grau para o qual uma interface de usuário permite uma interação agradável e satisfatória para o usuário.
- **Acessibilidade:** Grau em que um produto ou sistema pode ser usado por pessoas com a mais ampla gama de características e capacidades para atingir um objetivo especificado em um contexto especificado de uso.

### Confiabilidade

É representada pelo grau para o qual um sistema, produto ou componente executa funções especificadas sob condições especificadas por um período especificado. Essa característica é composta das seguintes subcaracterísticas [2]:

- **Maturidade:** Grau para o qual um sistema, produto ou componente atende às necessidades de confiabilidade sob operação normal.
- **Disponibilidade:** Grau em que um sistema, produto ou componente está operacional e acessível quando necessário para uso.
- **Tolerância ao erro:** Grau para o qual um sistema, produto ou componente opera conforme pretendido, apesar da presença de falhas de hardware ou software.
- **Recuperabilidade:** Grau para o qual, no caso de uma interrupção ou falha, um produto ou sistema pode recuperar os dados diretamente afetados e restabelecer o estado desejado do sistema.

### Segurança

É representada pelo grau, em que um produto ou sistema protege as informações e os dados para que as pessoas ou outros produtos ou sistemas tenham o grau de acesso aos dados adequado aos seus tipos e níveis de autorização. Essa característica é composta das seguintes subcaracterísticas [2]:

- **Confidencialidade:** Grau para o qual um produto ou sistema garante que os dados sejam acessíveis somente àqueles autorizados a ter acesso.
- **Integridade:** Grau para o qual um sistema, produto ou componente impede o acesso não autorizado ou a modificação de programas de computador ou dados.

- **Não repúdio:** Grau em que ações ou eventos podem ser provados como tendo ocorrido, de modo que os eventos ou ações não possam ser repudiados mais tarde.
- **Prestação de contas:** Grau para o qual as ações de uma entidade podem ser rastreadas exclusivamente para a entidade.
- **Autenticidade:** Grau em que a identidade de um sujeito ou recurso pode ser provada como sendo aquela reivindicada.

### Manutenibilidade

Esta característica é representada pelo grau de sua eficácia e eficiência com o qual um produto ou sistema pode ser modificado para melhorá-lo, corrigi-lo ou adaptá-lo a mudanças no ambiente e nos requisitos. Essa característica é composta das seguintes subcaracterísticas [2]:

- **Modularidade:** Grau em que um sistema ou programa de computador é composto de componentes discretos, de modo que uma alteração em um componente tenha impacto mínimo em outros componentes.
- **Reutilização:** Grau para o qual um ativo pode ser usado em mais de um sistema ou na construção de outros ativos.
- **Analisabilidade:** Grau de eficácia e eficiência com o qual é possível avaliar o impacto em um produto ou sistema de uma mudança pretendida para uma ou mais de suas partes, ou para diagnosticar um produto por deficiências ou causas de falhas, ou para identificar peças a serem modificadas.
- **Modificabilidade:** Grau para o qual um produto ou sistema pode ser efetivamente e eficientemente modificado sem introduzir defeitos ou degradar a qualidade do produto existente.
- **Testabilidade:** Grau de eficácia e eficiência com o qual critérios de teste podem ser estabelecidos para um sistema, produto ou componente e testes podem ser realizados para determinar se esses critérios foram atendidos.

### Portabilidade

Esta característica é representada pelo grau de eficácia e eficiência com o qual um sistema, produto ou componente pode ser transferido de um hardware, software ou outro ambiente operacional ou de uso para outro. Essa característica é composta das seguintes subcaracterísticas [2]:

- **Capacidade para ser Instalado:** Grau para o qual um produto ou sistema pode ser efetivamente e eficientemente adaptado para hardware, software ou outros ambientes operacionais ou de uso diferentes ou em evolução.
- **Coexistência :** Grau de eficácia e eficiência com o qual um produto ou sistema pode ser instalado e / ou desinstalado com êxito em um ambiente especificado.
- **Capacidade para Substituir:** Grau em que um produto pode substituir outro produto de software especificado para o mesmo propósito no mesmo ambiente

A Qualidade em Uso visa, cobrir não apenas a facilidade de uso, mas assegurar funcionalidades e suporte apropriado para atividades de uso em cenário real. É considerado não somente a visão do usuário, mas do contexto de uso em ambiente de trabalho [2]. A Figura2.9 apresenta as características de qualidade em uso.


Figura 2.9: Características de Qualidade em Uso
Fonte: Adaptado [2]

Conforme apresentado na Figura 2.9, a seguir segue as características de qualidade em uso [2]:

- **Eficácia:** Refere-se à capacidade do produto de software em possibilitar aos usuários atingir metas especificadas com precisão e integridade.
- **Eficiência:** Refere-se à capacidade do produto de software em possibilitar aos usuários atingir metas especificadas com completeza, num contexto de uso especificado.
- **Satisfação:** Refere-se à capacidade do produto de software em satisfazer usuários, num contexto de uso especificado.
- **Ausência de Risco:** Refere-se à capacidade do produto de software mitigar o risco potencial para a situação econômica, vida humana, saúde ou meio ambiente e a imagem.

- **Cobertura de contexto:** Refere-se à capacidade do produto de software pode ser utilizado com eficácia, eficiência, isenção de erros e satisfação em contextos específicos de uso e em contextos além daqueles inicialmente explicitamente.

Dentre os modelos apresentados nesta seção, será utilizado como base desta pesquisa a ISO25010 [2], pois segundo Miguel 2014[66], apresenta uma maior cobertura dentre todos os modelos dispostos na literatura, contando com uma maior quantidade de características para avaliação da qualidade, além de ter constante evolução em seu conteúdo, tendo sua última versão disponibilizada em 2011. Serão consideradas todas as características de qualidade para a identificação e análise de riscos, proposta na ferramenta.

A qualidade de um produto de software está fortemente relacionada à satisfação do cliente, e o fase de teste é a forma de validar se o produto desenvolvido atende às necessidades dos usuários [67]. Neste sentido, a fase de teste de software, tem um papel fundamental a será abordado na seção 2.3.

## 2.3 Teste de Software

O teste é uma disciplina da engenharia de software, e é considerado uma estratégia para gerenciamento de riscos, utilizada para verificar a conformidade dos requisitos com o software [37].

De acordo com Myers[16], teste é o processo de executar um programa com intuito de identificar defeitos. Já para Bach[21], teste é o processo de utilizar um software objetivando o encontro de falhas. Um caso de teste de sucesso é aquele que possui grande possibilidade de que falhas não encontradas anteriormente sejam reveladas, antes da entrega do produto. Neste sentido, são vários os conceitos em relação ao teste de software e às atividades do processo de teste, podem ser realizadas já nas fases iniciais do desenvolvimento, antes mesmo da codificação, através do planejamento e projeção dos casos de teste.

### 2.3.1 Conceitos Básicos

A execução de testes é utilizada para avaliar a qualidade de um produto e através de relatórios de resultado da execução dos casos de testes e rastreamento de requisitos funcionais e não funcionais, é possível obter percentuais de testes que estão passando, falhando ou que não puderam ser executados por meio de um conjunto de requisitos, em uma determinada versão, como forma de mensurar a cobertura dos testes [68].

Neste sentido um teste projetado adequadamente e cuja execução encontre defeitos reduz o nível de riscos em um produto de software, ser disponibilizado com baixa qua-

lidade. Por outro lado, quando os testes encontram defeitos, a qualidade do sistema aumenta quando estes são corrigidos. É importante destacar que testes mal projetados, ou executados de forma incorreta, podem encontrar poucos defeitos, deixando uma falsa impressão de qualidade [68].

Dessa forma, a fim de que se tenha entendimento sobre teste de software é necessário que alguns conceitos sejam bem entendidos, conforme apresentados[68]:

**Defeito (*Bug* ou falha):** Se um defeito no código for executado, o sistema falhará ao tentar fazer o que deveria (ou, em algumas vezes, o que não deveria), causando uma falha. Defeitos no software, sistemas ou documentos resultam em falhas, mas nem todos os defeitos causam falhas.

**Testes funcionais**: Identifica na realização de todos os níveis de teste, que são baseados na especificação, entende-se como teste de caixa preta.

**Testes não funcionais:** Podem ser realizados em todos os níveis de teste e são baseados na estrutura do código fonte, entende-se como teste de caixa branca.

Neste mesmo contexto, alguns princípios foram sugeridos pelo *syllabus International Software Testing Qualifications Board*, oferecendo um guia geral para o processo de teste como um todo citegraham2008foundations :

- **Teste demonstra a presença de defeitos:** O teste pode demonstrar a presença de defeitos, mas não pode provar que eles não existem. O teste reduz a probabilidade que os defeitos permaneçam em um software, mas mesmo se nenhum defeito for encontrado, não prova que ele esteja perfeito.
- **Teste exaustivo é impossível:** Testar tudo (todas as combinações de entradas e pré-condições) não é viável, exceto para casos triviais. Em vez do teste exaustivo, riscos e prioridades são levados em consideração para dar foco aos esforços de teste.
- **Teste antecipado:** A atividade de teste deve começar o mais breve possível no ciclo de desenvolvimento do software.
- **Agrupamento de defeitos:** Um número pequeno de módulos contém a maioria dos defeitos descobertos durante o teste antes de sua entrega.
- **Paradoxo do Pesticida:** Pode ocorrer de um mesmo conjunto de testes que são repetidos várias vezes não encontrarem novos defeitos após um determinado momento. Para superar este “paradoxo do pesticida”, os casos de testes necessitam ser frequentemente revisado e atualizado. Um conjunto de testes novo e diferente precisa ser escrito para exercitar diferentes partes do software ou sistema com objetivo de aumentar a possibilidade de encontrar mais erros.

- **Teste depende do contexto:** Testes são realizados de forma diferente conforme a necessidade. Por exemplo, software de segurança crítica são testados diferentemente de um software de comércio eletrônico.
- **A ilusão da ausência de erro:** Encontrar e consertar defeitos não ajuda se o sistema construído não atende às expectativas e necessidades dos usuários.

### 2.3.2 Modelo V

O ciclo de vida de testes foi concebido a fim de que todas as atividades de teste sejam realizados ao longo de todo o processo de construção [67]. Os ciclos de vida de testes e desenvolvimento são totalmente interdependentes, mas o ciclo de teste é dependente da conclusão dos produtos das atividades do ciclo de desenvolvimento. A seção 2.1.2 apresenta, por meio da Figura 2.2, como ocorre a iteração entre o processo de desenvolvimento e processo de teste. O vínculo entre os lados esquerdo e direito do modelo em V implica que, caso sejam encontrados problemas durante a verificação e a validação, o lado esquerdo do V pode ser executado novamente para corrigir e melhorar os requisitos, o projeto e a codificação, antes da execução das etapas de testes que estão no lado direito.

Existem duas divisões nas atividades de teste, conhecidas como validação e verificação (VV), e são compostas por um conjunto de atividades que são iniciadas em conjunto com a revisão dos requisitos, da análise, do projeto e pela inspeção do código fonte até os testes, conforme [67]:

- Verificação: revisão de requisitos, revisão de modelos, revisão de códigos e inspeções técnicas em geral;
- Validação: testes de integração, testes de software, teste unitários, testes de aceitação;

As atividades de VV são desenvolvidas por todas as etapas do processo de teste e do desenvolvimento do software, e mostra a relação entre os dois processos. No entanto, cada atividade possui características distintas entre elas, que são os testes dinâmicos e estáticos. Este modelo enfatiza a importância de considerar as atividades de testes durante o processo de desenvolvimento, ao invés de um teste posterior após o término do processo. Neste sentido, pode-se obter a retroalimentação mais rapidamente; ajuda a desenvolver novos requisitos; melhora a qualidade do produto resultante.

Considerando o exposto, este modelo será utilizado nesta pesquisa para demostrar a iteração entre os processos de desenvolvimento de software, de teste de software e processo de aplicação da ferramenta de Gestão de Risco para Qualidade de Software, apresentada nesta pesquisa.

### 2.3.3 Abordagens

No teste de software, na criação ou projeto dos casos são utilizados duas principais técnicas [68]:

- **Caixa Preta ou baseado em especificação:** São formas de derivar e selecionar as condições e casos de testes baseados na análise da documentação. Isto inclui testes funcionais e não funcionais para um componente ou sistema sem levar em consideração a sua estrutura interna. Ou seja, estão relacionadas às funcionalidades do sistema.
- **Caixa Branca ou baseadas em estrutura:** São baseadas na estrutura interna de um componente ou sistema. Ou seja, estão relacionadas à estrutura da codificação.

Os estágios definem o momento do ciclo de vida do software em que os testes são realizados e estágio de teste a ser realizado desde a elicitação de requisitos até o desenvolvimento do software, sendo eles testes estáticos ou dinâmicos. Os estágios de teste são [37]:

### 2.3.4 Estágios

- **Teste de Unidade:** Focaliza esforços de verificação na menor unidade de projeto de software ou componente de software. (ex: métodos e classes) são testados. Tem por objetivo testar a estrutura interna e comportamento do módulo e é geralmente realizado pelo desenvolvedor durante o desenvolvimento do software.
- **Teste de Integração:** É uma técnica sistemática para conduzir testes e descobrir erros associados às interfaces. Neste ponto, unidades testadas independentemente agora são testadas de forma integrada.
- **Teste de Sistema:** É uma série de diferentes testes cuja finalidade principal é exercitar por completo o sistema. Geralmente é um teste "caixa preta", executado por um testador de sistemas após a liberação de um executável do software. Abrange os seguintes tipos de teste: Teste Funcional, Teste Automatizado, Teste Exploratório, Teste de usabilidade, Teste de Acessibilidade.
- **Teste de Aceitação:** O software é testado pelo usuário final. É realizado um teste "caixa preta" a fim de demonstrar a conformidade com os requisitos de software.

A Figura 2.10, apresenta uma relação, dentro do Modelo V apresentado na seção 2.1.2, de quais tipos de teste devem ser aplicados dentro das fases de desenvolvimento de software.

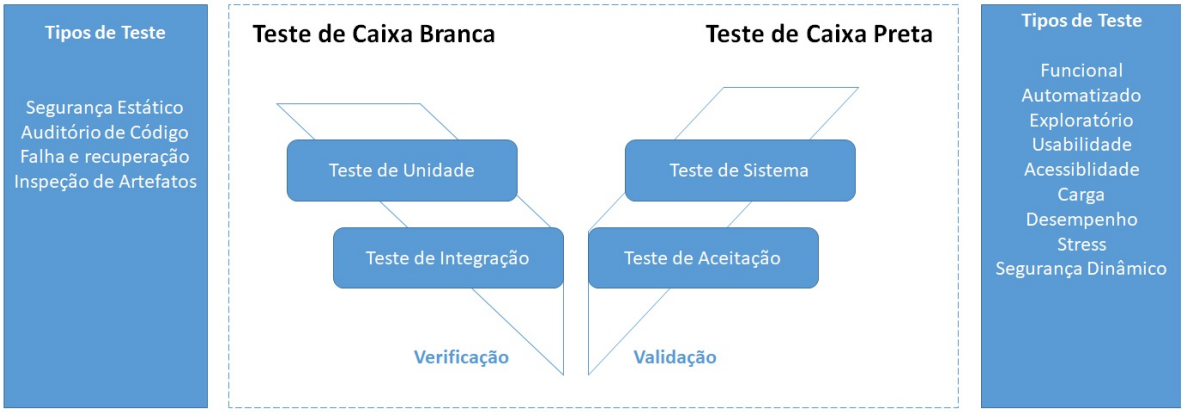



Figura 2.10: Relações dos tipos de teste no Modelo V
Fonte: Adaptado pela autora

### 2.3.5 Tipos de Testes

Com base nos conceitos apresentados, diferentes tipos de testes podem ser projetados e executados com o intuito de avaliar um objetivo específico. Neste ponto, serão apresentados os diversos tipos de testes e sua descrição na Tabela 2.1 [69]:

Tabela 2.1: Tipos de Teste

| Tipos de Teste | Descrição |
|---|---|
| **Teste Funcional** | O teste funcional de um sistema, envolve testes que avaliam as funções que o sistema deve executar. Os requisitos funcionais podem ser descritos em produtos de trabalho, como especificações de requisitos, de negócios, épicos, estórias de usuários, casos de uso ou especificações funcionais, podendo ainda não estarem documentados. As funções são "o que" o sistema deve fazer. |
| **Teste Automatizado** | Os testes automatizados tem como objetivo, exercitar o sistema, testando as funcionalidades e verificando se estão de acordo com as especificações dos requisitos do sistema e os objetivos esperados são desenvolvidos como programas ou scripts. |
| **Teste Exploratório** | Teste exploratório, normalmente o testador não tem informações detalhadas sobre o que vai testar e como vai testar. Quando se realiza um teste exploratório, normalmente o testador não tem informações detalhadas sobre o que vai testar e como vai testar. |
| **Teste de Segurança** | Verifica se os mecanismos de proteção incorporados ao sistema vão de fato protegê-lo de invasão imprópria. |
| **Auditoria de Código fonte** | Verifica se técnicas de programação foram utilizadas corretamente e identificar pontos com falhas. |
| **Teste de Usabilidade** | Verifica se a navegabilidade e os objetivos das telas funcionam como especificados |
| **Teste de Acessibilidade** | Verifica se o software está acessível a usuários com diferentes características. Ex. deficiente visual. |
| **Teste de carga, stress e performance** | Carga : Processo que testa e mede a alteração no desempenho da solução de software sob um volume maior de carga. Performance: Processo que testa e mede o desempenho da solução de software em uma situação normal de uso. Stress: Processo que busca descobrir qual a carga máxima suportada pela solução de software. |
| **Teste de falha e recuperação** | Verifica de o tempo exigido para recuperação do sistema caso falhe. |

Para esta pesquisa, o teste de software será utilizado como uma estratégia para o tratamento dos riscos. Uma vez que os riscos do produto de software sejam identificados,

os tipos de testes serão utilizados com o intuito de reduzir os riscos do não atendimento das características de qualidade do produto.

Neste sentido, para que seja possível aplicar as diversas abordagens de tipos de teste, é necessário que se tenha um processo de teste, para direcionamento nas atividades, conforme apresentado na seção 2.3.1.

### 2.3.6 Processo de Teste

De acordo com *International Software Testing Qualifications Board (ISTQB) [70]*, a parte mais visível do teste é a execução, entretanto para se obter eficácia e eficiência, os planos de teste precisam conter o tempo a ser gasto no planejamento dos testes, modelagem dos casos de testes e preparação da execução e avaliação de resultados. O processo de testes de software representa uma estruturação de fases, atividades que tem o objetivo de padronizar os trabalhos, além de maximizar a organização e monitoramento dos projetos de testes. Neste sentido segue o detalhamento do processo de teste, conforme apresentado na Figura 2.11 [70]:

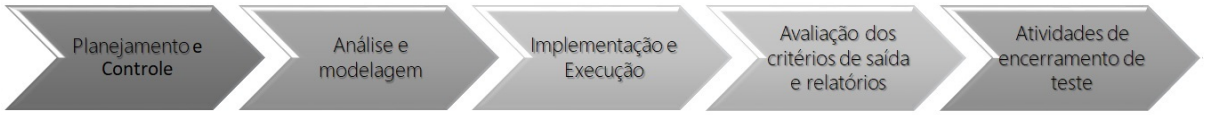



Figura 2.11: Processo de Teste
Fonte: Adaptado [70]

**Planejamento e controle:** O planejamento de teste é a atividade que consiste na definição dos objetivos e especificação das atividades de forma a alcançá-los e o controle do teste é a constante atividade que consiste em comparar o progresso atual contra o que foi planejado, reportando o status e os desvios do plano projetado.

**Análise e modelagem:** A análise e a modelagem de teste são atividades onde os objetivos gerais do teste são transformados em condições e modelos de teste tangíveis. Nesta etapa, realiza-se a projeção dos cenários de teste e se seguem as seguintes atividades:

- Revisar a base de teste (requisitos, integridade do software, nível de risco, arquitetura, modelagem e interfaces);
- Avaliar testabilidade
- Identificar e priorizar as condições
- Projetar e priorizar os casos de testes de alto nível.

- Identificar as necessidades de massa de teste, de acordo com as condições de testes projetadas
- Planejar a preparação do ambiente de teste e infraestrutura e ferramentas necessárias.
- Criar uma rastreabilidade entre os requisitos e os casos de teste.

**Implementação e execução:** A implementação e execução do teste é a atividade onde os procedimentos ou os *scripts* de teste são especificados pela combinação dos casos de teste em uma ordem particular, incluindo todas as outras informações necessárias para a execução do teste, o ambiente é preparado e os testes são executados. É orientado pelas tarefas a seguir:

- Finalizar, implementar e priorizar os casos de teste (incluindo a identificação de massa de teste).
- Projetar e priorizar os procedimentos de teste, criar massa de teste e caso de necessidade ao projeto, *scripts* de testes automatizados.
- Modelar/projetar suítes de teste a partir dos casos de teste para uma execução de teste eficiente. Este consiste de criar cenários de testes positivos e negativos para realização de testes.
- Verificar se o ambiente está preparado corretamente.
- Verificar e atualizar a rastreabilidade entre a base de teste e os casos de teste.
- Executar os casos de teste manualmente ou utilizando ferramentas de acordo com a sequência planejada na suíte de teste.
- Registrar e evidenciar os resultados da execução do teste e anotar as características e versões do software em teste,ferramenta de teste.
- Comparar resultados obtidos com os resultados esperados.
- Reportar as discrepâncias como incidentes e analisá-los a fim de estabelecer suas causas.
- Repetir os testes como resultado de ações tomadas para cada discrepância, ou seja, realizar um teste de confirmação, após a correção do incidente e ainda executar testes em áreas que sofreram impactos com as correções realizadas ou seja, teste de correção.

**Avaliação dos critérios de saída e relatórios:** A avaliação do critério de saída é a atividade onde a execução do teste é avaliada mediante os objetivos definidos.

- Verificar os registros de teste (*logs*) mediante o critério de encerramento especificado no planejamento de teste.
- Avaliar se são necessários testes adicionais ou se o critério de saída especificado deve ser alterado.
- Elaborar um relatório de teste resumido para as partes interessadas.

**Atividades de encerramento de teste** Na atividade de encerramento de teste são coletados os dados de todas as atividades para consolidar a experiência, fatos e números.

- Verificar quais artefatos a serem entregues
- Finalizar os relatórios de incidentes, ou levantar os registros de mudança que permaneceram abertos.
- Documentar o aceite do sistema.
- Realizar lições aprendidas para se determinar as mudanças necessárias para futuros *releases* e projetos.
- Utilizar as informações coletadas para melhorar a maturidade de teste.

O processo de teste fundamenta o direcionamento dos testes a serem realizados, a partir da identificação dos fatores de riscos do software. Desta forma, o teste apoia a aplicação do processo de gestão de risco no desenvolvimento de software, a fim de minimizar defeitos, conforme apresentado na seção 2.4.

## 2.4 Gestão de Riscos

A Gerência de Riscos pode ser estabelecida como o emprego de habilidades e competências aplicado ao conhecimento adquirido, por meio do uso de processos com a utilização de técnicas e métodos para a identificação, análise e controle dos riscos [13].

Considerando o exposto, serão abordadas a gestão de risco e suas características inerentes ao escopo desta pesquisa.

### 2.4.1 ABNT ISO 31000

A norma ABNT NBR ISO 31000: Gestão de riscos – Princípios e diretrizes define princípios e diretrizes em gestão de riscos e pode ser adotada por diferentes organizações nas atividades de decisão estratégica, operação, processo, função, projeto, serviço e avaliação de risco [1].

A gestão de riscos em sua forma mais ampla é definida como o conjunto de ações coordenadas para dirigir e controlar uma organização no que se refere a risco. Neste sentido, o risco pode ser definido como incerteza dos resultados. O propósito da gestão de risco é a criação e proteção de valor. Ela melhora o desempenho, encoraja a inovação e apoia o alcance de objetivos [1].

Neste contexto, a *framework* de gestão de riscos tem papel fundamental e consiste na definição de passos bem definidos para tomada de decisão na determinação das ações apropriadas para gerenciar os riscos em um nível aceitável pela organização. Recentemente foi publicado uma nova versão, a ABNT NBR ISO 31000:2018[1] esta pode ser aplicável em qualquer conjuntura, e permite os seguintes benefícios:

- Melhora proativamente a eficiência operacional e a governança;
- Constrói a confiança das partes interessadas na sua utilização de técnicas de risco;
- Aplica controles de sistema de gestão à análise de riscos para minimizar perdas;
- Melhora o desempenho e a resiliência do sistema de gestão;
- Responde às mudanças de forma eficaz e protege a sua empresa conforme ela cresce.

O processo de gestão de risco, envolve a aplicação sistemática de políticas, procedimentos e práticas para as atividades de comunicação e consulta, estabelecimento do contexto e avaliação, tratamento, monitoramento, análise crítica, registro e relato de riscos, conforme apresentado na Figura 2.12.

Um resumo será apresentado para cada etapa do processo da norma ABNT NBR ISO 31000:2018. Neste sentido a gestão de risco tem um papel iterativo, apesar de sua apresentação estar sequenciada [1]:

- **Comunicação e consulta:** O propósito da comunicação e consulta é auxiliar as partes interessadas na compreensão do risco, com base nas decisões que são tomadas e nas razões pelas quais ações específicas são requeridas. A comunicação busca promover a conscientização e o entendimento do risco e por sua vez a consulta envolve obter retorno e informação para apoiar na tomada de decisão.

  Envolve de uma forma ampla:

  - Reunir diferentes áreas de especialização para as etapas do processo de gestão de risco;
  - Assegurar que os pontos de vistas diferentes sejam considerados apropriadamente nos critérios de riscos e ao se avaliarem os riscos;


Figura 2.12: Processo ISO 31000
Fonte: [1]

- Minuciar de informações suficientes para facilitar a supervisão e a tomada de decisão;
- Incorporar um senso de inclusão e propriedade entre os afetados pelo risco;

- **Escopo, contexto e critérios:** Tem como objetivo personalizar o processo de gestão de risco ao contexto da organização, permitindo um processo de avaliação de risco eficaz e um tratamento de riscos apropriado no contexto interno e externo, conforme descrito a seguir:

  **Definição do escopo:** Estabelece o escopo das atividade de gestão de risco, que abrange:

  - Objetivos e decisões que precisam ser tomadas;
  - Resultados esperados do processo;
  - Tempo, localização, inclusão e exclusões específicas;

- Ferramentas e técnicas apropriadas para o processo de avaliação de riscos;
- Recursos, responsabilidades e registros a serem mantidos;
- Relacionamentos com outros projetos, processos e atividades;

**Contexto externo e interno:** Ambiente na qual a organização procura definir e alcançar seus objetivos. Convém o entendimento do contexto, visto que:

- A gestão de risco ocorre no contexto dos objetivos e atividades da organização;
- Fatores organizacionais, pode ser uma fonte de risco;
- Propósito e escopo do processo de gestão de riscos podem estar inter-relacionados no todo;

**Definindo critérios de risco:** Convém que a organização especifique a quantidade e o tipo de riscos que podem ou não assumir em relação aos objetivos. Além disso estabelecer critérios para avaliar a significância do risco e para apoiar os processos de tomada de decisão. Cabe ainda que os critérios de risco sejam estabelecidos no início do processo de avaliação, mas precisam ser continuamente analisados e para isso é necessário considerar:

- Natureza e o tipo de incertezas que podem afetar os resultados e objetivos;
- Como as consequências (tanto positivas quanto negativas) e as probabilidades serão definidas e medidas;
- Fatores relacionados ao tempo;
- Consistência no uso de medidas;
- Como o nível do risco determinado;
- Como as combinações e sequências de múltiplos riscos serão considerados;
- Capacidade da organização.

- **Processo de avaliação de risco:** Trata-se de um processo global de identificação de risco, análise de risco e avaliação de riscos. Convém que este processo de avaliação de risco, seja dirigido de forma sistemática, iterativa e colaborativa, com base no conhecimento e nos pontos de vista das partes interessadas.

  **Identificação do risco:** O Intuito da identificação de riscos é encontrar, reconhecer e descrever riscos que possam ajudar ou impedir que a organização alcance os objetivos. A organização pode utilizar várias técnicas para a identificação das incertezas que podem afetar um ou mais objetivo.

- Fonte tangíveis e intangíveis de riscos;
- Causas e Eventos;
- Ameaças e oportunidades;
- Vulnerabilidades e capacidades;
- Mudanças nos contextos externo e internos;
- Indicadores de riscos emergentes;
- Natureza e valor dos ativos e recursos;
- Consequências e seus impactos nos objetivos;
- Limitações de conhecimento e confiabilidade de informação;
- Fatores temporais e hipóteses e crenças dos envolvidos.

**Análise de Riscos:** O intuito da análise de riscos é compreender a natureza dos riscos e suas características, incluindo o nível de risco. Envolve a consideração detalhada de incertezas, fonte de risco, consequências, probabilidade, eventos, cenários, controles e sua eficácia.

As técnicas de análise de riscos podem ser qualitativas ou quantitativa, ou a combinação destas a depender do contexto. A seguir os fatores que precisam ser considerados:

- A probabilidade de eventos e consequências;
- A natureza e magnitude das consequências;
- Complexidade e conectividade;
- Fatores temporais e volatilidade;
- Sensibilidade e níveis de confiança.

### Avaliação de riscos

O intuito da avaliação de riscos é apoiar na decisão. A avaliação de riscos envolve o confronto dos resultados da análise de riscos com os critérios estabelecidos para determinar onde precisa de ação adicional.

- De nenhuma ação;
- Opções de tratamento de riscos;
- Realizar análise adicionais para compreender o risco;
- Manter os controles existentes;
- Reconsiderar os objetivos.

- **Tratamento de riscos:** Tem como intuito o tratamento do risco, ao qual se seleciona e implementa opções de abordar riscos. O tratamento do risco envolve um processo iterativo de:

  - Formular e selecionar opções para tratar o risco;
  - Planejar e implementar o tratamento do risco;
  - Avaliar a eficácia deste tratamento;
  - Decidir se o risco é remanescente é aceitável e se não for realizar tratamento adicional.

- **Monitoramento e Análise Crítica:** Tem como objetivo a análise crítica, com intuito de melhorar a qualidade e eficácia da concepção, implementação e resultados do processo. Convém um monitoramento contínuo e análise periódica do processo de gestão de risco e seus resultados.

- **Registro e relato:** Tem como objetivo a documentação dos relatos de risco, por meio de mecanismos apropriados.

  - Comunicar atividades e resultados de gestão d e risco em toda organização
  - Fornecer informações para tomada de decisão;
  - Melhorar as atividades de gestão de risco;
  - Auxiliar a interação com as partes interessadas.

O relato é parte integrante da governança da organização e tem como objetivo a melhora da qualidade do diálogo com as partes interessadas e apoio a alta administração.

As etapas apresentadas, serão utilizados como base para as fases de aplicação da ferramenta proposta nesta pesquisa.

Neste sentido a ISO31010, aborda ferramentas que apoiam em todo o processo de gestão de riscos, conforme apresentado na próxima seção.

### 2.4.2 Ferramentas de Gestão de Risco

Considerando o contexto deste trabalho, serão abordadas as principais ferramentas que são abordadas na ISO 31010 gestão de riscos [71].

**Brainstorming, Entrevistas estruturadas ou semiestruturadas:** Um meio de coletar um amplo conjunto de ideias e avaliação, classificando-o por uma equipe. O brainstorming pode ser estimulado através de instruções ou por técnicas de entrevista.

**Delphi:** Um meio de combinar opiniões de especialistas que possam apoiar a fonte e influenciar a estimativa de identificação, probabilidade e consequência e a avaliação de riscos. É uma técnica colaboradora para a construção de consenso entre os especialistas.

***SWIFT - Structured What If Technique:*** Um sistema para solicitar uma equipe para identificar os riscos. Normalmente é utilizada dentro de um workshop facilitado. Normalmente associada a uma técnica de análise e avaliação de riscos.

**Análise de cenário:** Aborda a possibilidade de possíveis cenários futuros. Podem ser identificados através da imaginação ou extrapolação dos riscos atuais e diferentes considerados, presumindo que cada um desses cenários pode ocorrer. Isto pode ser feito formal ou informalmente, qualitativamente ou quantitativamente.

**Análise de impacto nos negócios:** Provê uma análise de como os principais riscos de quebra, podem afetar as operações de uma organização e identifica e quantifica que capacidades que seriam requeridas para gerenciá-los.

As ferramentas apresentadas, fornecem princípios e diretrizes para gerenciar qualquer forma de risco de maneira sistemática, transparente e confiável dentro de qualquer escopo e contexto. Para a pesquisa, foram utilizadas como base como apoio de técnicas, para as fases de aplicação da ferramenta de Gestão de Riscos para Qualidade de Software, conforme apresentado na seção na seção 4.3.

Tendo em vista que a indústria de software traz consigo as suas particularidades, será abordado na seção 2.5 o risco no contexto específico desta pesquisa.

## 2.5 Risco na Engenharia de Software

Por meio das perspectivas globais de negócios, muitas empresas vêm tornando-se dependentes do sucesso ou do fracasso dos softwares que desenvolvem. Nesse sentido, a gerência de riscos não é apenas baseada em boas práticas para o desenvolvimento de software, mas sim boas práticas para gerir negócios [72].

De acordo com Pinna & Carvalho, riscos não gerenciados de forma adequada na engenharia de software podem comprometer a qualidade do produto final, a expectativa do cliente pode não ser atendida e a equipe, que precisa conviver com ansiedades e conflitos durante a vida do projeto, pode ter sua produtividade reduzida [73].

A área de Engenharia de Software tem promovido vários estudos com a finalidade de produzir modelos de melhoria, processo, métodos e ferramentas para aumentar a probabilidade de sucesso na execução de projetos de software, garantindo a qualidade dos seus produtos e minimizando possíveis riscos associados [74].

Vale destacar a diferença entre risco e problema. Os riscos são incertezas de que um evento futuro poderá afetar de forma negativa os objetivos do projeto, enquanto um

problema é algo que existe naquele momento e ameaça diretamente os objetivos do projeto. Ou seja, o problema é um risco já materializado. Esta diferença é importante, pois ações para gerenciar riscos serão muito diferentes das ações para gerenciar problemas. Como existe apenas a probabilidade de que o risco venha a ocorrer, ações podem ser tomadas para minimizar, transferir ou até eliminá-lo, ou seja, ações são realizadas preventivamente para que o risco não venha a se tornar um problema [75].

Muitas abordagens para gerenciar riscos em projetos de software vêm sendo propostas. Boehm em seu modelo espiral, apresentado na seção 2.1.3 [76] conseguiu trazer a atenção da comunidade de Engenharia de software para a necessidade de gerir risco, através de suas propostas de processos da gerência de risco.

A gestão de risco em Engenharia de software tem o intuito de aumentar a qualidade do produto e do processo de desenvolvimento de software. São várias as abordagens do processo de Gerência de Risco na engenharia. Embora tenham características próprias, cada abordagem tem alguns princípios e atividades em comum, conforme descritos a seguir:

Boehm[76] apresentou um processo para gerir riscos, envolve dois passos: Avaliação de Riscos (Identificação, Análise e Priorização de riscos) e Controle dos Riscos (Plano de gerenciamento de riscos, Resolução dos riscos e Monitoramento dos riscos)

Fairley[77] apresenta o processo de gerência de riscos em projetos de software através de sete passos: (1) Identificar os fatores de risco; (2) Avaliar as probabilidades e efeitos dos riscos; (3) Desenvolver estratégias para mitigar os riscos identificados; (4) Monitorar os fatores de risco; (5) Utilizar planos de contingência; (6) Gerenciar crises; (7) Sair de crises [6].

O Software Engineering Institute (SEI), através dos modelos do Capability Maturity Model Integrated (CMMI) [78], o processo de gerência de risco é composto por três fases: Avaliação de Riscos, Controle de Riscos e Relatórios de Riscos.

O Instituto de Engenharia de software (SEI), define risco como a possibilidade de sofrer perdas nos objetivos do projeto, como impactar na qualidade do produto final, atrasar cronograma, aumentar custos ou mesmo falhar o projeto. O processo de gerência de risco de software ocorre por meio de um modelo contínuo de gerenciamento de riscos composto por seis fases distintas: Identificação de Riscos, Análise de Riscos, Plano de respostas aos riscos, Rastreamento de riscos e Controle de riscos. Todas as fases estão ligadas através dos esforços de comunicação das equipes envolvidas no processo [78].

No RUP[79]*Rational Unified Process*, o processo de gerência de riscos é apresentado baseado em suas fases de desenvolvimento do produto, de forma sistemática:

- **Concepção**: Com ênfase nos riscos dos requisitos de negócio.
- **Elaboração**: Com foco nos riscos técnicos de definição da arquitetura do software.

- **Construção**: Com foco no tratamento dos riscos lógicos envolvidos na construção do produto.
- **Transição** – Com foco nos riscos funcionais de utilização do software.

### 2.5.1 Exposição ao risco

A ponderação de um risco é avaliada a partir da combinação da probabilidade ou frequência de ocorrência e da magnitude das consequências ou impacto dessa ocorrência. Uma das métricas mais conhecidas e utilizadas (Boehm, 1991) [76] para calcular a exposição ao risco é apresentada na Equação 2.1, onde P(f) representa a probabilidade de ocorrência do risco em uma função e I(f) o impacto desta ocorrência na função. Os valores de P e I são números definidos dentro de uma escala.

$$P(f) * I(f) \tag{2.1}$$

A Figura 2.13 apresenta a matriz de probabilidade e impacto representada no PMBOK[80], que exibe uma escala de probabilidade e impacto de 0 (zero) a 5 (cinco).

| ESCALA | PROBABILIDADE | +/- IMPACTO SOBRE OBJETIVOS DO PROJETO | | |
|---|---|---|---|---|
| | | TEMPO | CUSTO | QUALIDADE |
| Muito alto | >70% | >6 meses | > US$ 5 milhões | Impacto muito significativo sobre a funcionalidade geral |
| Alto | 51-70% | 3-6 meses | US$ 1M-US$ 5M | Impacto significativo sobre a funcionalidade geral |
| Médio | 31-50% | 1-3 meses | US$ 501.000 - US$ 1 milhão | Algum impacto em áreas funcionais essenciais |
| Baixo | 11-30% | 1-4 semanas | US$ 100.000 - US$ 500.000 | Impacto secundário sobre a funcionalidade geral |
| Muito baixo | 1-10% | 1 semana | < US$ 100.000 | Impacto secundário sobre funções secundárias |
| Nulo | <1% | Sem mudança | Sem mudança | Nenhuma mudança em funcionalidade |

Figura 2.13: Matriz de Definição de Probabilidade e Impacto
Fonte: PMBOK [80]

A representação para definição de probabilidade e impacto utilizada para este trabalho levou em consideração a escala apresentada, considerando os aspectos de qualidade definido no PMBOK [80].

### 2.5.2 Classificação

Risco na área de software foi representado de forma sistemática por Boehm em 1998 [43], que tem como princípio ser incremental e dirigido à análise de riscos. Atualmente, a área

que trata de riscos na Engenharia de Software evoluiu, passando de uma análise dentro dos modelos para se tornar uma gerência que permeia todos os processos do ciclo de vida do software.

Os riscos podem ser categorizados [10]:

- **Riscos de Projeto de Software:** Define os parâmetros operacionais, organizacionais e contratuais de desenvolvimento de software;
- **Riscos de Processo de Software:** Relacionam-se os problemas técnicos e de gerenciamento. Nos procedimentos técnicos podem se encontrar riscos nas atividades de Análise de Requisitos, projeto, codificação e teste;
- **Riscos de Produto de Software**: Contém as características intermediárias e finais do produto. Estes tipos de riscos têm origem nos requisitos de estabilidade do produto, desempenho, complexidade de codificação e especificação de testes.

Nesta pesquisa, para a ferramenta proposta, serão considerados os riscos de produto de software, sendo a possibilidade de uma funcionalidade apresentar falhas quando em uso pelo usuário, resultando em um produto de baixa qualidade. Todas as metodologias apresentadas trabalham com risco na engenharia de software, entretanto nenhuma provém uma gestão de risco voltada à qualidade de software. Considerando este contexto, será apresentado na seção 2.6 o conceito da gestão de risco para teste de software.

## 2.6 Teste de Software Baseado em Risco

Conforme apresentado nas seções anteriores, a atividade de teste é uma atividade que está diretamente associado ao risco. Quando a equipe de teste não realiza o teste em alguma funcionalidade do software, por motivos variados tais como custo, tempo e possibilidade de teste, estão assumindo riscos. Caso o evento de risco se materialize, o impacto dos riscos pode ser imensurável.

Em 1995, na revista *American Programmer*, através do artigo *The Challenge of Good Enough Software*, James Bach, apresentou uma abordagem de teste baseado em riscos, o que fez com que ele ficasse conhecido como o pai dessa abordagem [20].

A abordagem de teste de baseado em risco, consiste em um conjunto de atividades que favorece a identificação de fatores de riscos associados aos requisitos. Uma vez identificados, os riscos são priorizados, de acordo com sua probabilidade de ocorrência e impacto, assim o planejamento e projeção dos casos de testes, são elaborados com base na estratégia de tratamento dos fatores de riscos identificados. Quanto maior o risco, mais teste é necessário. Dessa forma, os defeitos com maior severidade podem ser descobertos mais cedo,

tendo em vista que os requisitos mais problemáticos serão testados prioritariamente, e consequentemente serão corrigidos mais cedo. Assim a execução dos casos de testes segue a priorização estabelecida na análise de risco e permite a mitigação dos riscos.

De acordo com Bach [21] o *Risk Based Testing*, tem como objetivo examinar:

- A cobertura dos testes;
- O número de testes a serem conduzidos;
- A escolha dos tipos de testes e revisões;
- O uso e o balanceamento entre os testes, as inspeções e as revisões;
- A priorização dos testes, o planejamento e a execução

Entretanto a avaliação e controle desses fatores de riscos não são atividades triviais e exigem bastante experiência. Não é simples saber como o produto pode falhar e determinar o quão importante as falhas seriam se ocorressem. Este processo se torna mais difícil quando praticado no ambiente típico de desenvolvimento, com pressão de prazos, custos, falta de documentação e partes interessadas diferentes sem conhecimento sobre gestão de risco e negócio [14, 22].

Neste sentido, focar em testes baseados em risco implica em fazer o julgamento sobre a cobertura de testes, seleção dos casos de testes a serem realizados, a seleção das técnicas e tipos de teste, fazendo um balanceamento entre os testes estáticos e dinâmico, entre outros problemas [22].

Vale ressaltar que os riscos tratados nesta abordagem são os riscos relacionados à engenharia de produto ou riscos técnicos, que podem ser mitigados através do teste de software.

A abordagem de teste baseado em risco, é fortemente utilizada, no planejamento e na estratégia de execução dos casos de teste [14]. Entretanto, seu uso também é recomendado na fase de modelagem de caso de teste, de forma a otimizar o processo, projetando apenas os casos de testes, prioritários, de acordo com a análise de risco realizada.

Bach [21] divide a abordagem em três etapas essenciais, conforme apresentado na Figura 2.14. Trata-se dos riscos técnicos do produto associados aos requisitos de software, aonde eles são identificados, analisados e controlados [21].

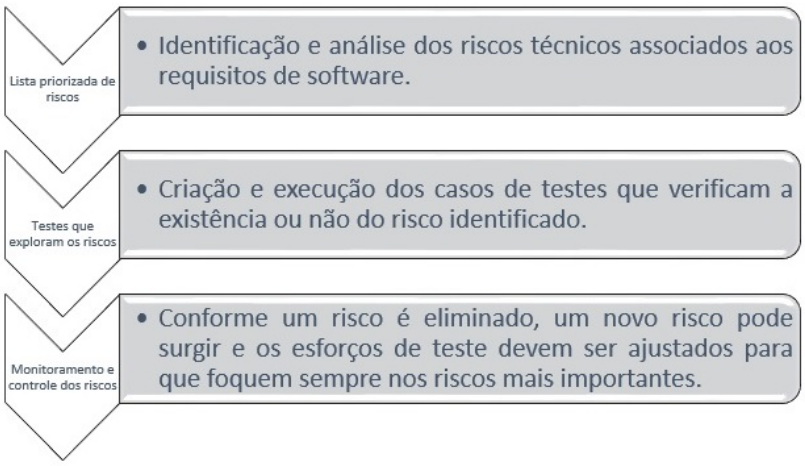



Figura 2.14: Etapas do teste baseado em riscos
Fonte: Adaptado pela autora [45]

As atividades do processo da gerência de risco [45], são apresentados na Figura 2.15 e foi criado com intuito de incluir as fases de teste junto ao processo de gestão de riscos. As elipses representam as alterações realizadas por Amland e são os artefatos produzidos ou atividades realizadas a partir dos elementos de retângulos que representam o processo de gestão de risco.

Neste sentido, para cada atividade do ciclo de vida do processo de teste de software há um correspondente na gestão de risco, com intuito de fornecer o tratamento adequado para os riscos técnicos identificados, conforme apresentado na 2.15.

Essa abordagem enfoca as atividades apresentadas na Figura 2.15, conforme detalhamento a seguir:

- **Identificação dos Riscos**

  A atividade de identificação de risco, responde às seguintes perguntas:

  - Existem riscos para esta função ou atividade?

  - Como pode ser classificado?

  A identificação de riscos envolve coletar informações sobre o produto e classificá-lo para determinar risco potencial na fase de teste e na produção.

- **Estratégia dos Riscos**

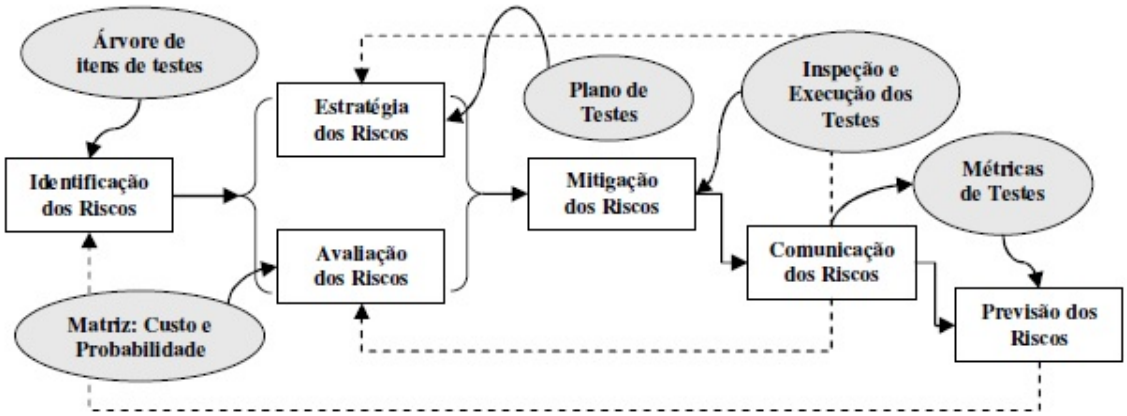



Figura 2.15: Processo de teste associado ao processo de gestão de risco
Fonte: Adaptado [45]

A estratégia e o planejamento baseados em riscos envolvem a identificação e avaliação de riscos e o desenvolvimento de planos de contingência para possível atividade de projeto alternativa ou a mitigação de todos os riscos. Este planejamento é utilizado para direcionar o gerenciamento de riscos durante as atividades de teste de software.

- **Avaliação dos Riscos**

  A Avaliação dos riscos, abrange determinar os efeitos de riscos potenciais. Avaliações de risco envolve os seguintes questionamentos:

  - Isso é um risco ou não?

  - Quão sério é o risco?

  - Quais são as consequências?

  - O que é a probabilidade de este risco acontecer?

  As decisões são tomadas com base no risco que está sendo avaliado. As decisões pode ser mitigar, gerenciar ou ignorar.

- **Mitigação dos Riscos**

  A atividade de mitigar e evitar riscos é baseada em informações obtidas nas etapas anteriores de identificação, planejamento e avaliação de riscos. As atividades evitam riscos ou minimizam impacto. A ideia é usar testes nas funções críticas para minimizar o impacto de uma falha esta função terá em produção.

- **Comunicação dos Riscos**

A comunicação dos riscos é baseado em informações obtidas dos tópicos anteriores, identificação, planejamento, avaliação e mitigação dos riscos. este relatório muitas vezes é representado por uma matriz com 4 quadrantes.

- **Previsão dos Risco**

  A previsão de risco é derivada das atividades anteriores e representa a previsão de risco, envolvendo o conhecimento adquirido sobre os riscos já identificados. Durante a execução do teste, é importante monitorar a qualidade de cada função individual (número de erros encontrado), para incluir testes adicionais ou mesmo rejeitar a função e enviá-lo de volta para o desenvolvimento se a qualidade for inaceitável. Esta é uma atividade contínua durante toda a fase de teste.

Este processo está alinhado ao processo de gestão de risco da ISO31000[1], entretanto adaptado ao contexto de teste de software. Considerando o cenário apresentado, a abordagem se mostrou eficiente à problemática da pesquisa, visto ser focado em teste de software, atividade esta que tem como objetivo a melhoria da qualidade dos sistemas e será utilizada como base para a construção da ferramenta proposta na pesquisa. Neste sentido, são várias as abordagens de teste baseado em risco disponíveis na literatura, e cada uma aborda uma técnica diferente. Serão apresentadas as técnicas e suas devidas características, na seção 2.6.1 à 2.6.6 para embasamento e captação de requisitos que poderão compor a ferramenta proposta nesta pesquisa.

### 2.6.1 Abordagem baseada em Heurística

Bach [21] criou duas abordagens distintas para identificação de riscos a serem explorados, baseadas em heurísticas, ou seja, baseadas em experiências anteriores. A primeira é *Inside-out*, ou seja, de dentro para fora e a segunda é *Outside-in*, ou seja, de fora para dentro. É importante ressaltar que dentre todas as abordagens de *Risk-based Testing* desenvolvidas, o estudo de Bach é o único que leva em consideração não só a análise de riscos que será alvo dos testes, mas também a maneira como o esforço de teste deve se organizar. Cada abordagem contém suas próprias características, mas podem se complementar.

***Inside-Out***

Esta abordagem consiste em analisar o produto do software de dentro para fora e se questionar "O que pode dar errado aqui?", levantando para cada componente ou funcionalidade as seguintes questões [21]:

- **Vulnerabilidades** – Quais fraquezas ou possíveis falhas podem ocorrer?
- **Ameaças** - Que entradas ou situações poderiam existir que podem explorar uma vulnerabilidade e disparar uma falha neste componente?

- **Vítimas**: Quem ou o que poderia ser impactado por uma possível falha e quão ruim isso seria?

A partir destas informações, os casos de teste são projetados. O levantamento das vítimas não auxilia a criação dos casos de teste, ele só ajuda a entender o quão grave foi a falha para uma determinada vítima e assim julgar se é necessário ou não escrever um caso de teste para aquela situação.

Importante ressaltar que a técnica prevê uma conversa informal entre o testador e desenvolvedor, com vistas a certificar opinião do desenvolver sobre o produto, aos pontos apresentados nesta técnica. É uma conversa não obrigatória, porém, nos casos que não ocorra, é importante que o testador estude, modele e analise o sistema sozinho.

***Outside –in***

Esta abordagem consiste em, a partir de um lista de riscos em potencial, relacioná-los aos detalhes de cada situação. Uma lista pré-definida de riscos é consultada e logo em seguida determinam-se quais riscos são aplicáveis naquele momento e naquela situação. Esta lista pode ser baseada em experiência própria ou em listas já existentes. Bach sugere três tipos de listas [21]:

**Lista baseado em modelos de qualidade**: São explorados a partir de uma lista baseadas nos modelos de qualidade ISO9126, atualizada por meio da ISO25010, FURPS[65, 61], entre outros. Para cada atributo de qualidade da tabela a seguir, é feito a seguinte pergunta: O que aconteceria se os critérios associados a estes atributos não forem cumpridos?

Tabela 2.2: Lista de categorias de critérios de qualidade

| | |
|---|---|
| **Capacidade** | O requisito realiza a função requerida? |
| **Confiabilidade** | As funcionalidades resiste a falhas em todas as situações ? |
| **Compatibilidade** | Quão bem a funcionalidade trabalha com componentes externos e outras configurações? |
| **Usabilidade** | Quão fácil a utilização da funcionalidade pelo usuário? |
| **Performance** | Quão rápido é o tempo de resposta da funcionalidade? |
| **Instabilidade** | Quão fácil é a instalação do software? |
| **Suportabilidade** | Quão econômica é a funcionalidade para fornecer suporte ao usuário? |
| **Testabilidade** | Como o produto pode ser testado efetivamente? |
| **Manutenibilidade** | Quão econômica são as manutenções, corretivas e evolutivas do produto? |
| **Portabilidade** | Quão econômica é a portabilidade ou reutilização para outra plataforma? |
| **Localizabilidade** | Quão econômica é a disponibilidade da funcionalidade em outra língua? |

**Lista Genérica de Riscos**: essa lista é baseado em algum modelo de qualidade e pode ser aplicado a qualquer sistema, conforme apresenta Tabela 2.2

**Catálogos de Risco**: são listas de riscos que pertencem a um domínio em particular. Alguns exemplos do que seria um catálogo de risco para um instalador. Instalação dos arquivos errados, Outras aplicações corrompidas, Não há detecção de aplicações incompatíveis, O processo de instalação é muito, O processo de instalação é confuso.

Estas técnicas prever questionamentos junto à área de infraestrutura e desenvolvimento.

Bach [21], também propõe, três maneiras de organizar os testes tendo por base os riscos, sendo elas:

- A lista de observação, consiste em uma lista de riscos que você periodicamente revê e se pergunta o que os testes revelaram sobre os mesmos.
- A matriz risco/tarefa, consiste em uma tabela cuja primeira coluna possui uma lista de riscos, organizados por importância, e a segunda coluna possui uma lista de tarefas de mitigação associada a cada risco.
- A matriz componente/risco é uma tabela com três colunas. A primeira indica o componente do sistema, a segunda, apresenta o grau de severidade do risco e a última a(s) heurística(s) que expõem o risco.

Entretanto, a principal dificuldade dessa abordagem está no conhecimento prévio do negócio, a fim de realizar a análise dos riscos da maneira mais assertiva. São grandes as possibilidades de a análise ter sido realizada de forma incorreta e consequentemente os riscos não terem sido atacados da forma correta. Além disso, segundo próprio Bach[21] a abordagem concentra-se em destacar os riscos que aumenta a necessidade de teste, mais não se concentram nos que diminuem a necessidade de testar. Ter claro os fatores que não necessitam tanto mas de teste seriam interessantes, mas para Bacth, poderia deixar a matriz complicada.

Dessa forma, a presente pesquisa utiliza técnicas apresentadas, com aplicação de melhorias dentre as falhas apresentadas por Bach[21]. A ferramenta proposta não foca na diminuição dos tipos de teste, mais sim em quais tipos de testes realizar de forma mais assertiva. Considerando este cenário, vamos explorar mais técnicas que utilizam a abordagem de teste baseado em risco apresentado na seção 2.6.2.

### 2.6.2 Abordagem baseada em Métricas

Essa abordagem, consiste em um conjunto de métricas desenvolvidas por Amland [23] para análise de risco com objetivo de auxiliar o processo de teste. Neste contexto as métricas propostas foram aplicadas por meio de um estudo de caso em uma organização financeira. O modelo é baseado na probabilidade de uma falha ocorrer e no custo dessa falha na função correspondente, tanto para o provedor do serviço quanto para o cliente.

A metodologia aborda as seguintes fontes de análise de risco:

**Qualidade da função a ser testada:** São levados em consideração aspectos como qualidade do produto de software, experiência dos desenvolvedores e complexidade, tamanho e maturidade das funcionalidades. De acordo com o autor, as funcionalidades complexas, de má qualidade e/ou desenvolvidas por desenvolvedores inexperientes estão mais expostas a falhas que funções baseadas em projeto de melhor qualidade e que foram desenvolvidas por programadores mais experientes. Essa função corresponde à probabilidade P(f), que pode assumir valores entre 0 e 1.

**Impactos de uma falha ponto de vista de um cliente:** Impactos no seu ambiente produtivo podem ser representadas, dentre outras, pela probabilidade de ameaça e perda de posicionamento no mercado e pelo não cumprimento de regulamentações governamentais. Estes impactos, representam o custo de uma falha para o consumidor C(c).

O impacto de uma falha do ponto de vista do vendedor do serviço está relacionado, dentre outras, à probabilidade de divulgação negativa, altos custos de manutenção de software e perda de clientes. Estes impactos representam o custo de uma falha para o vendedor C(v).

Para Anland, o custo para o consumidor é igualmente importante na análise dos riscos em relação ao custo do vendedor. Dessa forma, a exposição de risco é dada pela seguinte função:

$$\mathrm{Re}(f) = P(f) * \frac{C(c) + C(v)}{2}$$

Figura 2.16: Cálculo para exposição do risco
Fonte: [23]

De acordo com o grau de exposição ao risco, as áreas com maior risco, tem maior prioridade nos testes e, dessa forma, durante a execução dos testes, à medida que falhas vão sendo apresentadas, novas prioridades podem ser adicionadas ao projeto. Neste sentido, conforme o grau de exposição ao risco vai sendo atualizado para cada funcionalidade, a prioridade do teste pode ir sendo modificada.

Além das métricas para o cálculo de exposição ao risco, Amland também utiliza, em seu estudo de caso, métricas para acompanhamento do progresso dos testes, sendo eles número de casos de testes planejados e executados, número de defeitos por funções, quantidade de horas gastas em teste por defeito encontrado e quantidade de horas gastas para correção de defeitos.

A abordagem utiliza as atividades, que foram apresentadas na Figura 2.15:

**Planejamento:** Realização do planejamento do teste, com a definição de quais funcionalidades serão testadas, padrões de documentação, ambiente de teste, execução e registro dos incidentes. Neste contexto a estratégia de riscos também é definida.

**Identificar os indicadores de riscos**: Realizar a definição junto à equipe de projetos. Os indicadores definidos precisam fazer sentido para o sistema e todos devem ter o mesmo entendimento. Podem ser utilizados indicadores de linhas de códigos e mudanças nas funções, número de defeitos, entre outros.

**Identificar o custo de um defeito**: Da mesma forma da atividade listada, os indicadores são definidos com foco no custo. Neste sentido, pode ser utilizado o custo de um defeito, tanto para o usuário (cliente), quanto para o desenvolvedor (vendedor) do produto de software.

**Identificar elementos críticos**: Realizar os cálculos para a exposição do risco de cada funcionalidade utilizando os valores obtidos nas atividades de Identificação de indicadores de risco e de custo de defeito.

**Execução dos testes**: Após a conclusão da análise dos riscos, os testes seguem a ordem de execução da lista de priorização e as dependências entre os casos de testes e adaptações necessárias.

**Estimar o tempo para completar**: Com base nas métricas de progresso dos testes, reportar a situação atual dos testes e prever o tempo necessário para completá-los.

Essa abordagem foi aplicada através de um estudo de caso em uma instituição financeira e um dos maiores problemas relatados por Amland foi falta de conhecimento de algumas funcionalidades, que dificultou a análise e produziu resultados não confiáveis. Mesmo com as dificuldades apresentadas, foi projetado um nível mínimo de teste para funcionalidades com baixa exposição ao risco e testes extras foram definidos para funcionalidades com maior exposição ao risco. Ao final, o cliente avaliou o produto entregue com excelente qualidade. O número de defeitos encontrados foi similar ao das versões anteriores, porém o tempo gasto para concluir os testes e o número de recursos utilizados foi menor. O grande desafio da definição da abordagem, segundo Amland, foi a identificação dos indicadores de custo de falha e de qualidade. Foi utilizada uma abordagem simples, mas, segundo o autor, satisfatória.

Desta forma, o autor mantém foco somente na atividade de análise dos riscos, não fornecendo subsídios para a identificação dos riscos associados aos requisitos, fundamental para a criação dos casos de teste baseado em riscos. A abordagem baseada em código-fonte, apresenta uma visão baseada em métricas, entretanto voltado à código fonte, conforme apresentado na seção 2.6.3.

### 2.6.3 Abordagem baseada em Código-fonte Orientado a Objetos

Trata-se de uma metodologia fornecida por Rosenberg [24], que se apoia na identificação das classes de código fonte que estejam propensas a erros, através de métricas de complexidade ciclomática, ou seja, métricas de complexidade de código. Esse método de teste baseado em risco leva em consideração a probabilidade de falha de uma parte código, de acordo com a determinação da sua complexidade, a partir de códigos fonte orientados a objeto.

Refere-se a uma técnica de teste altamente eficaz que pode ser utilizada para localizar e corrigir os problemas mais importantes o mais rapidamente possível. O risco pode ser caracterizado pela combinação de dois fatores, sendo o primeiro, pela gravidade de um potencial evento de falha e o segundo pela probabilidade da sua ocorrência.

O teste baseado em risco concentra-se na análise do software e na realização de um plano de teste ponderado às áreas com maior probabilidade de sofrer problemas que teriam maior probabilidade de impacto [25].

Segundo Pfleege[81],códigos mais complexos, têm maiores chances de apresentar erros ou problemas. Considerando este ponto, seis métricas de medição de projetos orientadas a objeto são utilizadas, identificadas e aplicadas pelo software Assurance Technology Center ( SATC) da Goddard Space Flight Center na NASA, sendo elas[82, 83]:

- **Número de Métodos ou *Number of Methods*(NOM)**: é uma contagem simples dos diferentes métodos em uma classe.
- **Número Ponderado de Métodos Por Classe ou *The Weighted Methods per Class*(WMC):** soma ponderada dos métodos em uma classe. Se os pesos forem iguais, equivale à métrica anterior. A complexidade ciclomática é usada para avaliar a número mínimo de casos de teste necessários para cada método.
- **Acoplamento entre objetos ou *Coupling Between Objects (CBO)*:** é uma contagem do número de outras classes nas quais uma classe está acoplada
- **A resposta a uma classe ou *The Response for a Class (RFC):*** é a cardinalidade do conjunto de todos os métodos que podem ser convocados em resposta à uma mensagem para um objeto da classe.
- **Profundidade na árvore ou *Depth in Tree (DIT)*:** A profundidade de uma classe referente a sua hierarquia de herança é a número de saltos da classe para a raiz da hierarquia de classes e é medido por o número de classes ancestrais. Quando existe múltipla, heranças, use-se o máximo DIT.
- **Número de filhos ou *Number of Children (NOC)*:** O número de filhos é o número de subclasses que herdam diretamente da classe na hierarquia.

A empresa SATC, por mais de três anos, coletou dados e analisou códigos orientados nas linguagens Java e C++. Foram analisados mais de 20.000 classes de mais de 15 software e discutidas com gerentes de projetos e desenvolvedores para identificação dos valores limites. Com base nesta análise, foi possível chegar a uma distribuição das métricas coletadas, conforme apresentada a Tabela 2.3.

Tabela 2.3: Métricas para valores individuais
Fonte[24]

| METRICA | VALOR LIMITE |
|---|---|
| **NOM** | Preferencialmente menor que 20 e aceitável até 40 |
| **WMC** | Preferencialmente menor que 25 e aceitável até 40 |
| **CBO** | Aceitável até 5 |
| **RFC** | Aceitável até 50 |
| **DIT** | Aceitável até 5 |
| **NOC** | Não há conceso, mas quanto maior, mais alta a probabilidade de erro |

Para avaliação dos riscos do código, são necessárias, pelo menos, duas ou três métricas para termos indicação de um problema em potencial. O alto risco é identificado para classes que tem, ao menos, duas métricas que excedem os limites recomendados.

O objetivo das métricas coletadas é identificar as classes com maior risco de erro. Embora não haja dados suficientes para fazer determinações precisas de classificação, há informações para justificar testes adicionais de classes que excedam as métricas definidas. Neste contexto, cabe ao projeto determinar a criticidade destes e das outras classes para fazer o planejamento final no teste. Dessa forma, alocar esforços de teste com base nesses dois fatores (gravidade e probabilidade de falhas) equivale a testes baseados em riscos nesta abordagem.

Existem ferramentas que se apoiam na análise das classes, um exemplo delas é o SonarQube [84].

No entanto os autores não propõem estratégias de testes para o código, tampouco formas de rastreamento das funcionalidades impactadas pelas classes mapeadas com alto risco de falhas[24]. Dessa forma, a abordagem baseada em regressão, consegue este rastreamento, e será explicado na seção 2.6.4.

### 2.6.4 Abordagem para Teste de Regressão

A abordagem, introduz uma metodologia baseada em risco para testes de regressão[26].

O objetivo principal do teste de regressão é a garantia de que mesmo após modificações, o sistema continue estável. Através dos testes de regressão queremos alcançar confiança

Esta abordagem define dois conjuntos de testes de regressão:

- **Testes de regressão para as funcionalidades que foram alterados:** pelo menos um teste é executado para cada funcionalidade inserida ou alterada;
- **Testes de regressão baseado em Riscos para os funcionalidades que "aparentemente" não sofreram alterações:** para estes, uma análise de riscos é realizada a partir de questionários submetidos aos participantes do projeto.

Esta abordagem modela os casos de teste que focam nas áreas do software que possuem maior risco. Como efeito, apoiar no atingimento do nível de confiabilidade adequado na qualidade do software. Esta abordagem também utiliza o modelo de gestão de risco, apresentado na Figura 2.15 e envolve os seguintes fases [85]:

- Estimar o custo de cada caso de teste. Ou seja, o custo que uma falha possa causar.
- Priorizar da severidade dos caso de teste. Esse valor é computado a partir do número de defeitos e da severidade destes defeitos.
- Calcular o grau do risco para cada caso de teste. A exposição ao risco é calculada a partir do custo e da severidade.
- Selecionar os casos de teste que têm os maiores valores de exposição ao risco.

A seleção de cenários é realizada da soma da exposição ao risco dos casos de testes associados ao cenário e, a partir desta soma, devem ser selecionados os cenários com maior exposição ao risco. Os cenários são uma sequência de passos, descrevendo a interação entre usuário e o sistema. Após a isso, uma matriz de rastreabilidade é feita, mostrando quais casos de teste de cada cenário é coberto. Este processo pode ser realizado quantas vezes for necessário. Neste sentido, a seleção dos cenários baseada em riscos deve seguir as seguintes regras:

**Regra 1:** Selecionar os cenários para cobrir os casos de teste mais críticos;

**Regra 2:** Garantir que os cenários cubram maior quantidade de casos de teste quanto possível

A abordagem foi aplicada por meio de um estudo de caso e se mostrou bastante eficiente. Além disso, a análise realizada através de questionários submetidos aos desenvolvedores, com questões que eles dominam, não constituiu uma tarefa complexa. Segundo a autora, com a ajuda de ferramentas, todo o processo foi realizado de forma rápida. Como a abordagem toma como base a documentação do sistema, esta, por sua vez, deve encontrar-se sempre atualizada. Nesse ponto, surge uma abordagem baseada na experiência do usuário e em sua utilização, que será explicado na próxima seção.

### 2.6.5 Abordagem baseada em Uso

Esta abordagem, descreve como fazer testes baseados em riscos, priorizando as funções vitais do sistema, de acordo com Besson [27].

O método inicia-se controlando o esforço de teste, baseado na utilização do sistema de acordo com a teoria de Pareto que afirma que 20% das funcionalidades permitem aos usuários realizar 80% do seu trabalho. Neste sentido o autor sugere testar apenas as

funcionalidades incluídas nos vinte por cento, desta forma reduzindo os esforços e custos dos testes e aumentando a qualidade.

O método é iniciado na identificação das funcionalidades vitais do sistema, que podem prevenir o usuário de utilizar o sistema. A gravidade é definida medindo o impacto negativo que uma falha tem sobre o negócio (alta, média ou baixa). Neste sentido, identificar funcionalidades que podem prevenir o usuário de utilizar o sistema se um defeito for encontrado, ou seja, um defeito com alta severidade.

A forma de obter essas informações é através de pesquisa com o usuário final e perguntar aos especialistas do domínio ou usar estatísticas do log de versões anteriores do sistema na identificação das funções mais utilizadas, visto que essas aumentam o risco. Após isso, é necessário a projeção dos casos de teste e estimativa do tamanho através do esforço necessário para executar os casos de testes identificados e logo após executá-los.

Esse método seleciona apenas funções com alta severidade e classifica os casos de teste de acordo para o tempo necessário para executar cada caso de teste, ou seja, a execução de testes baseada em esforço, os testes que gastam menos tempo são executados primeiro até que o tempo acabe. Uma priorização entre as funcionalidades selecionadas com base na gravidade. Entretanto o autor expõe que o método “é possivelmente falho", pois não leva em conta a dependência do caso de teste. Se os casos de testes são classificados apenas no tempo, então pode encontrar dependência entre casos de teste que podem exigir mais esforços.

Considerando a situação apresentada, esta abordagem é relativamente fácil de ser aplicada, uma vez que o usuário especifica as funcionalidades que são mais utilizadas no sistema, entretanto a sua cobertura em relação aos testes é baixa. Além disso, essa abordagem pode ser útil em culturas organizacionais avessas ao teste, visto que se testa somente 20% das funcionalidades disponíveis no software.

### 2.6.6 Prisma

Trata-se de um método para identificação das áreas mais importantes para testar, através dos itens que tem maior nível de risco. O método PRISMA suporta o gerenciador de testes ao executar testes, especialmente para a identificação e análise de riscos em estreita cooperação com partes interessadas[28]. Neste sentido, análise de risco do produto deve ser usada para determinar a abordagem de teste apropriada e selecionar técnicas de projeto de teste de tal forma que os itens com os maiores riscos são testados primeiro e mais intensamente do que as áreas com baixo risco. Este método utiliza o conceito de "Bom o suficiente", que significa que já existe o suficiente deste produto funcionando para que possamos levá-lo à produção, usá-lo, obter valor e obter benefício, não tendo problemas críticos. Ou seja, que não há falhas graves que o tornam inutilizável ou inaceitável [28].

Para o método, nem todos os produtos precisam estar livres de defeitos, ou seja, você pode reduzir os testes em áreas menos importantes. Uma maneira de garantir isso é encontrar as mais importantes áreas funcionais e propriedades do produto. Encontrar tantos defeitos quanto possível pode ser melhorado testando mais nas áreas ruins do produto. Desta forma, é possível tomar a decisão sobre o que testar e o que não testar, o que testar mais ou o que testar menos [28].

Neste método, foi criado um processo e uma ferramenta, que tem como intuito principal a criação da matriz de risco do produto, conforme descrito a seguir [28]:

**Planejamento:** Nesta fase é identificado todos os documentos de entrada. Idealmente os documentos a serem utilizados são os documentos de requisitos ou documento de arquitetura do sistema e que são entradas para a identificação dos itens de riscos, além de entrevistas junto aos criadores dos documentos e ou partes interessadas. Neste sentido, a lista de itens de riscos deve ser identificada e compreensível de forma única por todos os participantes. Após isso, deve-se atribuir pesos aos atributos, podendo receber os valores de 1, 2 e 3, sendo que 1 "fator não é muito importante", 2 para "fator tem influência normal ", 3 para" fator tem forte influência". Uma vez definidos os pesos, as partes interessadas são selecionadas, envolvendo gerente de projetos, desenvolvedores, arquiteto de softwares, marketing, usuário final, área de negócio e engenheiro de aplicação. Normalmente, os fatores de impacto devem ser atribuídos aos gestores de negócio e fatores de probabilidade para especialistas técnicos.

***Kick-off:*** Trata-se de uma fase opcional, que pode ser realizada em uma reunião em que o gerente de teste explica a todas as partes interessadas o papel de cada um no processo. Embora opcional, a reunião de kick-off é altamente recomendada. Pode ser utilizada também para explicar a lista de itens de risco e esclarecer dúvidas. Os itens e fatores de risco são explicados em detalhes, já que às partes interessadas serão solicitadas que as pontue.

**Preparação Individual:** Nesta fase, os valores são atribuídos aos fatores por item de risco pelos participantes. Eles pontuam selecionando o valor que melhor se ajusta ao risco percebido para o fator correspondente em relação a um item de risco.

**Reunir pontuações individuais:** Durante a coleta de pontuações individuais, o gerente de teste verifica primeiro se a pontuação foi feita corretamente. Se ainda houver violações às regras, o gerente de teste deve discutir isso com o participante. Logo após, é realizado o processamento das pontuações individuais calculando a média do valor, além de preparar uma lista de questões a serem discutidas na reunião de consenso.

**Reunião de consenso:** A reunião de consenso é iniciada pelo gerente de teste explicando os objetivos da reunião. No final desta reunião, um entendimento comum deve ser alcançado para os riscos de produto. O resultado desta fase é uma matriz de risco compro-

metida pelas partes interessadas e aderindo ao conjunto de regras, conforme apresentado na Figura 2.17.

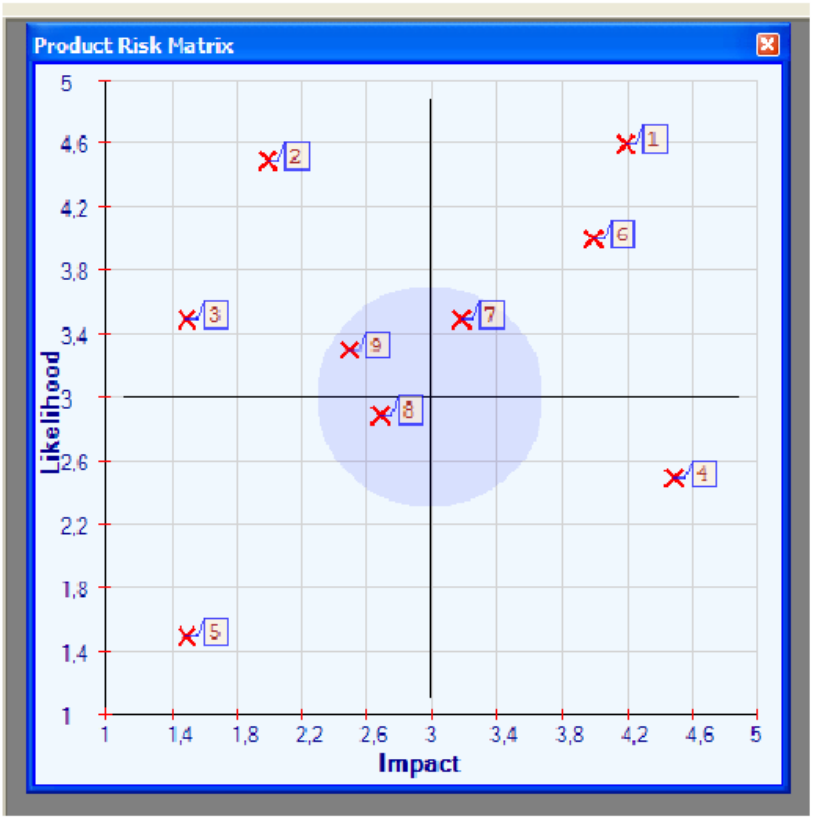



Figura 2.17: Exemplo de uma matriz de risco da ferramenta PRISMA
Fonte: [86]

Durante a reunião, os itens de riscos são discutidos e compreensões diferentes dos requisitos é fonte de mudança sobre os requisitos, ou seja, os itens de risco, uma vez que estes obviamente não são ambíguos. Se os resultados não forem de acordo com as expectativas das partes interessadas, elas devem ser rediscutidas.

**Definir uma abordagem de teste diferenciada:** Com base na localização dos itens de teste na matriz de risco, todos os itens de teste são priorizados. Além da priorização, uma abordagem de teste diferenciada para os itens do teste precisa ser definida com base em sua a matriz de risco. A abordagem de teste geralmente tem dois aspectos principais: a profundidade de teste a ser aplicado e a prioridades para testes. A profundidade do teste pode ser realizada de várias maneiras, utilizando diferentes técnicas de teste. Além de utilizar diferentes técnicas em um projeto de teste, existem alternativas para definir abordagem diferenciada com base na matriz de risco resultante. Práticas para considerar quais podem ser aplicados para definir uma abordagem diferenciada incluem testes estáticos, revisão de projetos de teste, testes de regressão, nível de independência

Tabela 2.4: Pesquisa de Utilidade

| **Critério** | **Pontuação média** |
|---|---|
| Útil | 7.5 |
| Eficiência | 7.2 |
| Eficácia | 7.0 |

e critérios de saída, como uma cobertura de declaração alvo. Também itens de alto risco podem ser testados pelos engenheiros mais experientes, outra forma de mitigar o risco.

O escopo, o contexto ou os requisitos dos projetos geralmente exigirão atualizações das etapas do processo de avaliação. O processo de avaliação de risco torna-se, portanto, iterativo.

De acordo com o estudo de caso apresentado por Erick [28], não foi possível realizar uma comparação entre testes de projetos usando o método PRISMA, entretanto foi realizado um questionário com vinte empresas que já haviam aplicado o PRISMA em vários de seus projetos de teste. O objetivo do estudo foi investigar se os gerentes de teste e testadores, consideram o método e a ferramenta como benéfica para a tarefa de detectar defeitos. O questionário foi aplicado considerando uma escala de 1 a 10, sendo que 1 é discordo totalmente e 10 fortemente aceita, conforme perguntas a seguir:

- Até que ponto o PRISMA é útil para executar a tarefa de gerenciamento de teste?
- Até que ponto o uso do PRISMA torna o processo de teste mais eficiente?
- Até que ponto o uso do PRISMA torna o processo de teste mais eficaz?
- Quão fácil é aplicar o PRISMA em um projeto da vida real?

A pesquisa de utilidade demonstra que a maioria dos respondentes consideram a abordagem PRISMA útil. A percepção dos gerentes de teste sobre os vários itens indica que o PRISMA aumenta a eficácia e eficiência das tarefas de detecção de defeitos. Isto confirma a suposição de que o PRISMA exibe uma relação positiva entre uso e desempenho.

Comentários interessantes feitos pelos participantes sobre a utilidade do PRISMA incluem:

- Apoia a tomada de decisões corretas quando o projeto está sob pressão;
- O risco é uma linguagem de negócios e, portanto, PRISMA é uma boa ferramenta para se comunicar com partes interessadas;
- Uma boa base para a estimação de testes e definição de uma abordagem de teste diferenciada;
- Fornece uma estrutura para monitoramento e controle de testes durante o projeto;

- Um foco claro para as atividades de teste, também já durante os testes de desenvolvimento;
- Garante que as partes mais importantes do produto sejam testadas antes do lançamento;
- Como resultado da avaliação de risco, o nível de prioridade dos defeitos encontrados aumentou;
- Uma base para aprender e conscientizar sobre os fatores que influenciam o risco, também para ser usado durante a melhoria do processo.

Já a pesquisa de facilidade de uso, apresenta resultados menores. Na prática, foi descoberto que é necessário um treinamento e consultoria para a aplicação do método, de acordo com os comentários listados:

- Definir os itens de risco de forma independentes no nível correto e agrupá-los em aproximadamente 30 itens é um desafio;
- Às vezes é difícil identificar as partes interessadas (de negócios) e obtê-las envolvidos, especialmente pela primeira vez;
- Para alguns interessados envolve uma mudança de mentalidade, eles agora mais explicitamente tornar-se donos do risco;
- Fazer escolhas explícitas é difícil para algumas partes interessadas (de negócios), elas sempre pensam que tudo foi testado "totalmente";
- A interpretação dos fatores não é fácil, um pontapé inicial é uma boa definição dos fatores (incluindo regras de pontuação) são altamente recomendados;
- O desenvolvimento nem sempre está alinhado com as prioridades de teste, os itens de risco mais importantes são entregues relativamente tarde no processo;
- Definir uma abordagem de teste diferenciada baseada nos riscos é difícil, depende também da conhecimento e nível de habilidade dos engenheiros de teste envolvidos;
- Finalmente, a maior parte da avaliação de risco baseia-se no risco percebido no início do projeto, entretanto os projetos tendem a ser dinâmicos e as pessoas aprendem ao longo do projeto a avaliação de risco também deve ser tratado desta maneira e deve ser repetido do método nos marcos de entrega dos projetos de teste.

O referencial teórico apresentado nesta seção, exibe as abordagens e ferramentas voltadas a teste de software baseado em risco, que foram base para elicitação de requisitos da ferramenta, conforme exibido no capítulo 4, da ferramenta proposta.

Finalizada a apresentação dos principais referenciais teóricos que deram subsídios para o desenvolvimento do trabalho, o capítulo 3 apresenta a metodologia que define as etapas da pesquisa.

# Capítulo 3

# Metodologia de Pesquisa

Esse capítulo apresenta a metodologia aplicada ao trabalho, onde serão demostrados os métodos utilizados e os procedimentos para o alcance do objetivo geral e de cada um dos objetivos específicos propostos nesta pesquisa.

## 3.1 Classificação da pesquisa

Método científico pode ser definido como um conjunto de etapas e instrumentos pelo qual o pesquisador pode direcionar o seu projeto com critérios de caráter científico para alcançar dados que suportam ou não sua teoria inicial [87].

Desta forma, uma etapa importante do processo é a definição e estruturação dos instrumentos que serão utilizados para cada tipo de pesquisa a fim de obter resultados confiáveis [88].

De acordo com Gil [89], a pesquisa é definida como um procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos. A pesquisa é desenvolvida por um processo constituído de várias fases, desde a formulação do problema até a apresentação e discussões de resultados.

Os tipos de pesquisa podem ser classificados quanto a sua Abordagem, Natureza, Objetivos e Estratégia. Neste sentido, a Figura 3.1 apresenta a estrutura desta pesquisa e métodos utilizados [89, 88].

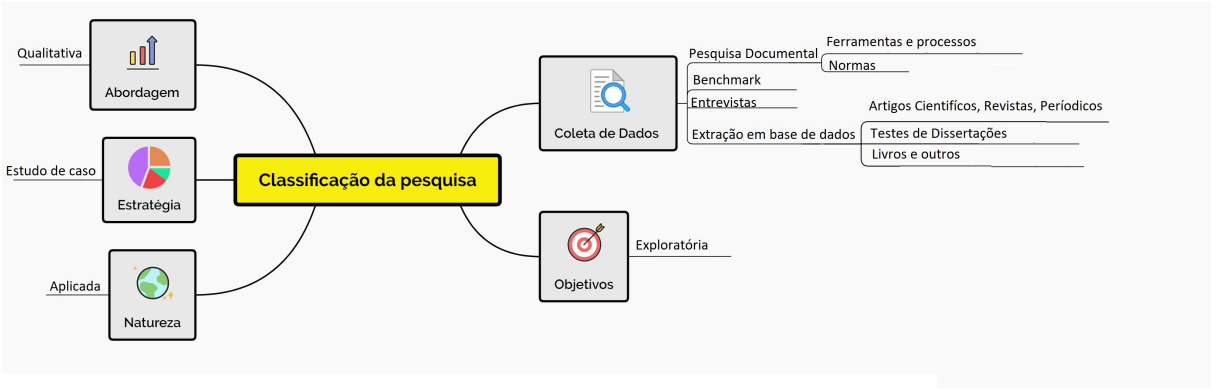



Figura 3.1: Classificação da Pesquisa

A seguir o detalhamento da pesquisa:

- Quanto ao Objetivo, a pesquisa foi de caráter exploratório utilizada com vistas a proporcionar uma maior familiaridade com o problema, deixando-o assim mais explícito. Considerando este item, foi realizado um levantamento bibliográfico, entrevista com especialistas no tema abordado e análise de exemplos que estimulem a compreensão.
- Quanto à Natureza, foi utilizada a pesquisa aplicada, que gerou conhecimento para aplicação prática, através do desenvolvimento da ferramenta que permitiu a gestão de riscos em ambiente de desenvolvimentos com foco na garantia da qualidade do produto, com intuito de melhorar a qualidade do produto de softwares desenvolvidos.
- Quanto à Abordagem, a pesquisa foi de caráter qualitativo, ao qual permitiu a descrição, compreensão, explicação da gestão de risco aplicada a qualidade e teste de softwares, bem como os resultados da análise comparativas com as ferramentas já existentes, e ainda com a proposta e validação da ferramenta proposta.
- Quanto à Estratégia, foram utilizados estudos de casos, pois, embora a ferramenta tenha propósito de generalização, ou seja, aplicada em qualquer contexto de desenvolvimento de software, para fins de validação, foi aplicada em dois projetos de desenvolvimento de softwares em metodologias distintas.

  Essa pesquisa situa-se no âmbito das áreas de estudo da Engenharia da Produção através da gestão de riscos e da computação aplicada por meio da Engenharia de Software.

## 3.2 Estrutura da Pesquisa

A pesquisa foi estruturada em 3 etapas, conforme apresentado na Figura 3.2 e detalhado a seguir:


Figura 3.2: Estruturação das etapas da pesquisa

**Levantamento das necessidades**

Esta etapa abrangeu o primeiro objetivo específico e foi iniciada por meio da revisão da literatura, que subsidiou o desenvolvimento da pesquisa através de referenciais teóricos e outras pesquisas relevantes. Abarcou a pesquisa das principais bases de dados, tais como *Web of Sciente*, *Scopus*, Google Acadêmico entre outros, por meio de artigos científicos nacionais e internacionais, livros, teses e dissertações sobre as metodologias, práticas, ferramentas e técnicas de análise de risco no processo de desenvolvimento de software, voltadas a qualidade do produto.

Foram levantadas informações sobre a engenharia de software e os modelos de desenvolvimento utilizados na construção de um software, para entendimento de como a fase de gestão de riscos e de teste de qualidade de software se interagem em seu ciclo de vida de desenvolvimento, a fim de que a ferramenta proposta possa ser aplicada a qualquer modelo de desenvolvimento. Abrangeu os principais modelos, incluindo o modelo cascata, modelo em V, modelo espiral, interativo e incremental, RUP e Ágil. Posteriormente fo-

ram pesquisados os conceitos de qualidade e quais modelos a literatura mais utiliza para avaliação de um software de qualidade. A pesquisa envolveu os principais modelos de qualidade, Mc-Call's Quality Model, Boehm's Quality Model, Dromey Quality Model e FURPS Quality Model, além das normas internacionais ISO 9126 e ISO25010. Adiante foi conceituado o teste de software e como se dá sua iteração entre os modelos de desenvolvimento e como, por meio do teste, pode-se buscar a qualidade dos softwares, através dos tipos de teste e seu processo. Em seguida, foi conceituado a gestão de risco, por meio da ISO31000 e 31010 e principais modelos de gestão de riscos aplicado a engenharia de software, sendo o SEI, CMMI e RUP.

Realizou-se um estudo e pesquisa sobre o teste de software baseado em risco, por meio das abordagens e ferramentas voltadas a este modelo, que associa a atividade de teste a gestão de risco, com vistas a buscar a qualidade dos softwares. A pesquisa abarcou a Abordagem Baseada em Heurística, em Métricas, em Teste de Regressão, Em Uso e a ferramenta PRISMA, que foram utilizados para uma análise comparativa e identificação de pontos positivos e negativos das atuais abordagens que deram base para a elicitação de requisitos da ferramenta proposta na pesquisa.

A revisão permeou, de forma recorrente, todas as etapas da pesquisa, e foi base para todo referencial deste trabalho.

### Desenvolvimento da ferramenta

Essa fase compreendeu o segundo objetivo específico da pesquisa e se deu pela composição da ferramenta, por meio das etapas de desenvolvimento de software no modelo iterativo incremental e incorporou as fases apresentadas na Figura 3.3, conforme detalhamento a seguir:

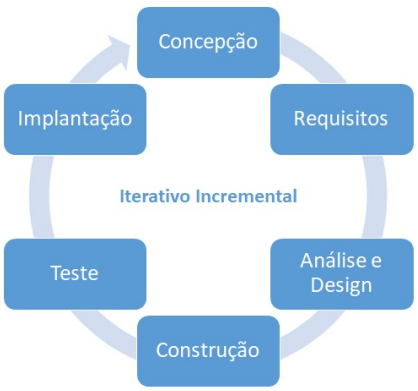



Figura 3.3: Fases de desenvolvimento da ferramenta de Gestão de Risco para Qualidade de Software

Fonte: Elaborado pela autora

- Concepção: Esta fase englobou a identificação das necessidades e oportunidades e tradução dessas em problemas e determinação dos objetivos e metas a serem alcançados, além da base conceitual para determinação do problema, conforme apresentado no Capítulo 1 e 4, seção 4.2.1.
- Requisitos: Esta fase incluiu a elicitação dos requisitos, por meio da fase de levantamento das necessidades e coleta de dados, além da documentação dos requisitos coletados para o desenvolvimento da ferramenta, apresentadas na seção 4.2.2.
- Análise de Desing: Esta fase incluiu o desenho da ferramenta, definição da arquitetura, a escolha das tecnologias e definição do MER - Modelo Entidade Relacionamento, que permitiu a descrição do modelo de dados para descrever os aspectos de informação de um domínio de negócio ou seus requisitos de processo, conforme apresentado na seção 4.2.3.
- Construção: Esta fase foi composta pela construção da ferramenta, ao qual foram aplicados as regras e requisitos descritos da seção 4.2.2 desta pesquisa de acordo com o detalhamento apresentação na seção 4.2.4.
- Teste: Esta fase foi composta pela validação do produto de software, com vistas a identificar defeitos por meio de realização de testes em cada funcionalidade, antes do produto ser disponibilizado ao seu ambiente produtivo, descrito na seção 4.2.5.
- Implantação: Esta fase compreendeu a disponibilização da ferramenta ao ambiente do usuário, via web. Abrangeu a construção de manual da ferramenta, disponibilizado no Apêndice A, desta pesquisa.

Atingido o objetivo específico 2, por meio do desenvolvimento da ferramenta de gestão de riscos técnicos do produto no ciclo de desenvolvimento, foi aplicado estudos de caso para validação da ferramenta proposta.

### Aplicação da Ferramenta

Para o terceiro objetivo específico foi aplicada a solução desenvolvida para o problema apresentado na pesquisa. Neste contexto, a ferramenta foi aplicada em projetos de Tecnologia da Informação (TI) em metodologias de desenvolvimento de software distintas, através de dois estudos de caso, sendo possível demonstrar se a ferramenta desenvolvida atendeu aos objetivos para os quais foram propostos. Foi, também, realizado um comparativo da qualidade do produto de software com a aplicação da ferramenta e sem a sua utilização, com intuito de medir a sua efetividade.

## 3.3 Técnicas para coletas de dados

As principais técnicas de coleta de dados utilizadas na pesquisa, foram Benchmarking, Reuniões, entrevistas não estruturadas e extração de dados de bases de dados, conforme detalhamento a seguir:

- *Benchmarking*: Realizado dentro da RSI Informática, empresa especialista em teste e qualidade de software junto aos clientes que utilizam algum tipo de ferramenta de análise de risco voltada para este fim. Neste sentido, dentre uma carteira de mais de 30 clientes ativos foram identificados 5 clientes, representando 16% da amostra que utilizam, de maneira mais sólida e incorporada ao processo de desenvolvimento de software, práticas de gestão de riscos voltadas a Qualidade de Software. As abordagens utilizadas também foram apoio para elicitação de requisitos da ferramenta proposta na pesquisa.

  Foi realizada uma visita aos clientes, ao qual os mesmos apresentaram as ferramentas que utilizam dentro de seus processos de desenvolvimento, voltadas a gestão de risco para Qualidade de Software. Após a reunião, o material utilizado, foi enviado a pesquisadora, para uma análise mais aprofundada as ferramentas utilizadas, conforme apresentado na Seção 4.2.2.

- *Entrevistas não estruturada*: Conduzidas pela pesquisadora e realizadas na empresa RSI Informática, junto a especialistas da área. A entrevista foi realizada com 8 especialistas dentre um corpo de 32 constante na empresa, representando um total dos 25% dos peritos, com o questionamento do que poderia compor a nova ferramenta. O resultado é apresentado na seção 4.2.1 desta pesquisa.

- Extração em base de dados ( *Web of Science, Scopus* e outros): teses de dissertações, para levantar conjunto de documentos e artigos relevantes e selecionar as informações importantes para compreensão do processo, assim como das ferramentas e técnicas utilizadas. A pesquisa abrangeu normas, processos de gestão de risco para produto de software, Risk Based Test (RBT) e manuais e tutoriais de ferramentas.

## 3.4 Principais ferramentas de software utilizadas

Nesta pesquisa, foram utilizadas várias ferramentas de software para apoio do desenvolvimento do processo e da ferramenta de gestão de risco com o foco na garantia da qualidade do produto, conforme descritas abaixo:

**Microsoft Excel:** Trata-se de uma poderosa ferramenta, de visualização e análise de dados e parte integrante de um software disponível em praticamente todos os computa-

dores e, portanto, acessível para a maioria das pessoas que é o *Microsoft Office*. Concerne de uma planilha eletrônica utilizada em atividade que implique o processamento de um grande volume de cálculos financeiros repetitivos ou simulação de situações envolvendo simultaneamente múltiplas variáveis e ainda geração de gráficos. O Microsoft Excel é um programa bastante amigável e, portanto, de rápida aprendizagem por qualquer pessoa. Desta maneira é uma ferramenta de fácil acesso que pode ser utilizada para resolver problemas matemáticos [90, 91]. Nesta pesquisa, foi utilizada para o desenvolvimento do protótipo, a fim de validar sua aplicabilidade e objetivo ao qual foi concebida.

***APEX - Oracle Application Express:*** Trata-se de uma ferramenta de desenvolvimento de aplicativos da Web rápida para o banco de dados Oracle. Permite, por meio de um navegador Web, o desenvolvimento de softwares profissionais rápidos e seguros. Permite a criação de interface do usuário, controles de navegação, manipuladores de formulários e relatórios flexíveis. Do ponto de vista do usuário final, os aplicativos implantados exigem apenas um navegador e acesso a um banco de dados Oracle que executa o *Application Express* [92]. Nesta pesquisa, foi utilizada a versão 9, para o desenvolvimento da Ferramenta de Gestão de Risco para Qualidade de software.

**Mantis:** Trata-se de uma ferramenta de código-aberto escrita na linguagem PHP. Tem como objetivo a gestão de defeitos. Permite o relato de defeitos e implementação do ciclo de vida de defeitos necessário para que todos os envolvidos no projeto de testes possam identificar em que estado encontra-se o defeito e para quem está atribuído. Foi utilizada para a gestão dos defeitos dos projetos de testes e apresentação dos indicadores aplicados nos estudos de caso desta pesquisa. Disponível para *download* em: https://www.mantisbt.org/.[93]

***UML - Unified Modeling Language:*** A crescente complexidade dos sistemas de informação vem desafiando a maneira como arquitetos e engenheiros de software trabalham. Depois de inicialmente estar mais preocupado com a estrutura e qualidade do código de programação, os engenheiros de software concentraram sua atenção nos aspectos de modelagem do sistema e processo de desenvolvimento [94]. Desta forma UML é uma linguagem gráfica universal e tornou-se um dos padrões mais utilizados para especificar e documentar sistemas de informação[94].

Os modelos fornecem abstrações de sistemas que ajudam a lidar com aplicativos maiores e mais complexos de formas mais simples, independentemente de como são implementadas e distribuído e qualquer plataforma de execução final ou tecnologia utilizada [94].

Neste sentido, a *Unified Modeling Language (UML)* foi utilizado para documentar o software desenvolvido nesta pesquisa, com vistas a representar a ferramenta por meio de um diagrama de caso de uso. Este diagrama, representa de forma visual o sistema

do ponto de vista do usuário e descreve as principais funcionalidades dos sistemas e a interação entre elas.

***PowerDesigner:*** Trata-se de uma ferramenta produzida pela empresa americana *Sybase* que permite aos usuários suportar algumas fases e tarefas de processo de desenvolvimento de software ou sistemas de informação. A ferramenta permite a realização de Modelagem de processos, Análise de Requisitos, Arquitetura de software, Arquitetura de Dados e Controle de Versão. A ferramenta trabalha com a notação estrutural e modelagem de interação em notação UML. Neste contexto, a ferramenta foi utilizada para a diagramação do Modelo Entidade Relacionamento (MER) da ferramenta desta pesquisa.[95]

***Power BI:*** O *Power BI* é uma ferramenta que fornece visualizações interativas e recursos de *Business Intelligence (BI)*, com uma interface simples para que os usuários finais criem os seus próprios relatórios e dashboards. Nesta pesquisa, foi utilizada para a apresentar os indicadores do Monitoramento e Análise Crítica [96].

***Trello*** Trata-se de uma ferramenta de gestão de projeto, bastante conhecido por ser uma ferramenta extremamente versátil e que pode ser ajustada de acordo com as necessidades do usuário. Trabalha por meio de um quadro kanban, totalmente customizado a sua necessidade [97]. Nesta pesquisa, foi utilizada para o controle dos testes da ferramenta desenvolvida, com intuito de controlar os defeitos e melhorias identificadas, por meio dos testes de software.

Finalizada a apresentação da metodologia da pesquisa, o Capítulo 4 apresenta a ferramenta de Gestão de Risco para Qualidade de software.

# Capítulo 4

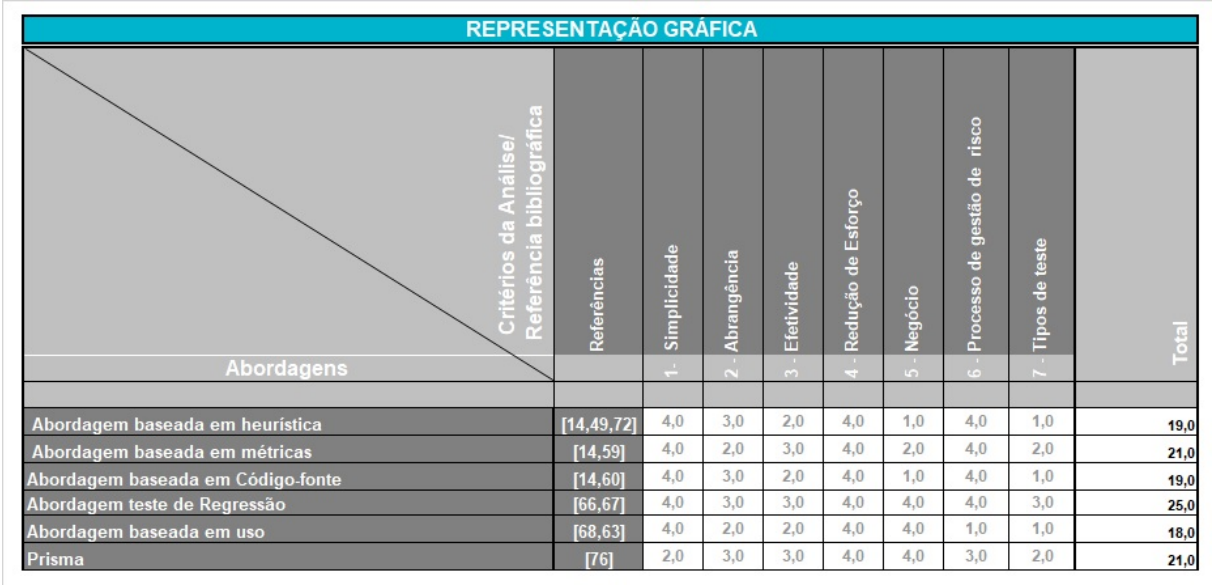
# Ferramenta de Gestão de Risco para Qualidade de Software

O objetivo deste capítulo é realizar a apresentação da ferramenta, expondo inicialmente uma pesquisa qualitativa, em uma análise comparativa dentre as abordagens e ferramentas já existentes; Posteriormente as fases de desenvolvimento para composição da nova ferramenta e sua estrutura arquitetural; logo em seguida a descrição de cada uma das fases de maneira detalhada, de como a ferramenta de Gestão de Risco para Qualidade de software foi concebida e ainda seu processo de aplicação. Apresenta-se ainda sua aplicação por meio das fases de desenvolvimento de software.

A ferramenta de Gestão de Risco para Qualidade de software foi desenvolvida com a finalidade de identificar os fatores de risco associados aos requisitos de forma simplificada nas fases de desenvolvimento, com intuito de concentrar os esforços nas funcionalidades mais críticas e ainda ter um estratégia de teste bem definida com vistas a reduzir a probabilidade e impacto da má qualidade de software em seu ambiente produtivo.

## 4.1 Pesquisa Qualitativa para Construção da Ferramenta

Considerando as abordagens de teste de software baseado em risco demonstradas na seção 2.6 do referencial teórico, foi realizado um estudo comparativo das ferramentas e abordagens atuais identificadas na literatura, para embasar a construção da nova ferramenta. Para a análise, foram definidos alguns critérios, para a avaliação das abordagens já existentes, conforme apresentado na Figura 4.1. Esses critérios foram pontuados, com base na tabela 4.1, para análise nas abordagens avaliadas.



REPRESENTAÇÃO GRÁFICA

| Critérios da Análise/ Referência bibliográfica / Abordagens | Referências | 1 - Simplicidade | 2 - Abrangência | 3 - Efetividade | 4 - Redução de Esforço | 5 - Negócio | 6 - Processo de gestão de risco | 7 - Tipos de teste | Total |
|---|---|---|---|---|---|---|---|---|---|
| Abordagem baseada em heurística | [14,49,72] | 4,0 | 3,0 | 2,0 | 4,0 | 1,0 | 4,0 | 1,0 | 19,0 |
| Abordagem baseada em métricas | [14,59] | 4,0 | 2,0 | 3,0 | 4,0 | 2,0 | 4,0 | 2,0 | 21,0 |
| Abordagem baseada em Código-fonte | [14,60] | 4,0 | 3,0 | 2,0 | 4,0 | 1,0 | 4,0 | 1,0 | 19,0 |
| Abordagem teste de Regressão | [66,67] | 4,0 | 3,0 | 3,0 | 4,0 | 4,0 | 4,0 | 3,0 | 25,0 |
| Abordagem baseada em uso | [68,63] | 4,0 | 2,0 | 2,0 | 4,0 | 4,0 | 1,0 | 1,0 | 18,0 |
| Prisma | [76] | 2,0 | 3,0 | 3,0 | 4,0 | 4,0 | 3,0 | 2,0 | 21,0 |

Figura 4.1: Representação das abordagens e ferramentas analisadas
Fonte: Elaborado pela autora

Os critérios definidos, foram elencados tendo como base os conceitos apresentados no teste de software baseado em risco, que define os critérios que precisam ser contemplados, para que se tenha resultados efetivos na aplicação da abordagem [20, 21, 14].

**Abrangência:** Foi analisado a abrangência da aplicação considerando a abordagem de teste baseado em risco, conforme apresentado na seção 2.6 deste trabalho [20, 21, 68]. Foram examinados a cobertura de testes, o número de testes a serem conduzidos, o balanceamento entre quais tipos de testes aplicar. A utilização dos conceitos de qualidade, também serão analisados de acordo com os modelos de qualidade dispostos nesta pesquisa [63, 61, 65, 2].

**Efetividade:** Foi apreciado a efetividade da aplicação da abordagem no que se refere a melhorar o nível de qualidade do produto de software e diminuição dos erros embasando a abordagem de teste baseado em risco [20], com vistas a garantir a cobertura de teste e qualidade do software em seu ambiente produtivo.

**Redução do esforço:** Foi analisado se a abordagem permite redução de esforços aliados a abrangência e efetividade. O intuito da nova ferramenta é que o esforço gasto nas atividades de teste e qualidade seja bem aproveitada através das priorizações [20, 39].

**Negócio**: Foi verificado o envolvimento da área de negócio na abordagem. O intuito é que a nova ferramenta tenha a percepção de qualidade dos gestores e usuários finais, com vistas a ser mais assertivo na análise de risco, visto que a própria abordagem de teste baseado em risco aborda a preocupação e necessidade da experiência no negócio para uma avaliação mais efetiva [14, 22].

**Processo de gestão de risco:** Foi analisado a aplicação do processo de gestão de risco e teste nas abordagens apresentadas. O intuito é que nova ferramenta de gestão de risco para qualidade de software esteja aliada ao processo de gestão de risco disposto na ISO31010 e ainda na Análise de Risco Baseado em Teste de Software [1, 45].

**Tipos de teste:** Foi analisada a escolha do tipo de teste e revisões necessárias para a mitigação, de acordo com o conceito do teste de software baseado em risco, que aborda a necessidade do balanceamento entre os testes, as inspeções e as revisões [20]. O objetivo é que nova ferramenta permita a seleção dos tipos de testes aliados aos riscos identificados, com vistas a aplicar as técnicas certas ao contexto apresentado.

Tabela 4.1: Tabela de Critérios

| **Itens da Avaliação** | **Nota** |
|---|---|
| A abordagem atende aos critérios de forma adequada. | **4** |
| A abordagem tem um bom atendimento aos critérios, podendo haver alguma restrição, porém de baixo impacto | **3** |
| A abordagem atende de forma razoável os critérios com restrições | **2** |
| A abordagem não atende aos critérios | **1** |

A tabela 4.1 foi construída, levantando em consideração a escala Likert [98], avaliando posicionamento negativo e positivo quanto a cobertura das abordagens e ferramenta, por meio dos estudos de casos. Para esta escala, foram utilizados apenas 4 itens, uma vez que a opção “Indiferente", não cabe para a avaliação. É descrita a seguir a relação entre a Tabela de Critérios e a Escala utilizada como referência.

A pontuação de cada abordagem aos seus critérios, foram realizadas considerando os resultados apresentados para cada estudo de caso. Desta forma, segue análise realizada para cada abordagem, em relação aos critérios elencados:

**Abordagem baseada em heurística:** Welter [99], realizou um estudo de caso em um ambiente virtual de aprendizado, utilizando esta abordagem. Neste sentido, a análise foi realizada com base neste estudo de caso, dissertação de mestrado de Souza [75], e artigo do próprio autor, com relação a abordagem analisada[21] logo abaixo:

- No critério simplicidade, o estudo de caso apresentou ser uma atividade de simples aplicação.
- No critério abrangência, foi demonstrado que a abordagem leva em consideração os conceitos de qualidade, através dos critérios da ISO9126 e FURGS e ainda permitiu agilidade, na execução de forma eficaz no ambiente testado, entretanto não define como os casos de testes são gerados e não sugere técnicas de tipo de teste.
- No critério efetividade, atende, entretanto não complemente, visto que o foco é testar as áreas com maior risco e não realizar testes em todas as funcionalidades.

- No critério redução de esforço, como sua aplicabilidade, foca nos testes das áreas com maiores riscos de falhas, acaba tornando a atividade de teste mais ágil e menos custosa.
- No critério negócio, foi apresentado dificuldade nessa abordagem, visto que é necessário o conhecimento prévio do negócio a fim de realizar a análise mais assertiva. Foi apresentado grandes possibilidades da análise ter sido feita de forma incorreta e consequentemente os riscos não serem atacadas de formas correta.
- No critério processo de gestão de risco, segue o processo de acordo com a Figura 2.15 e aborda a ISO31000 de gestão de risco.
- No critério tipos de teste, o autor, não sugere utilizar de acordo com os riscos identificados.

**Abordagem baseada em métricas:** O autor Amland, apresenta um estudo de caso em uma organização, que foi aplicada para testar dois módulos de uma sistema financeiro, contendo respectivamente doze e treze funcionalidades. Neste sentido, a análise desta abordagem foi realizada com ba se neste estudo [23] em outros estudos dispostos na literatura [75, 100]:

- No critério simplicidade, segundo o autor, foi utilizada uma abordagem simples e satisfatória.
- No critério abrangência, o autor mantém foco somente na atividade de análise dos riscos, não fornecendo subsídios para a identificação dos riscos associados aos requisitos, fundamental para a criação dos casos de teste baseado em riscos, entretanto o autor utiliza algumas métricas como por exemplo, número de casos de testes planejados e executados, número de defeitos por funções, número de horas gastas em teste por defeito encontrado e número de horas gastas para correção de defeitos.
- No critério efetividade, o autor relata que um nível mínimo de teste foi definido para as funcionalidades com baixa exposição ao risco e testes extras foram definidos para funcionalidades com maior exposição ao risco. Ao final o cliente avaliou o produto entregue com excelente qualidade. Neste sentido, o número de defeitos encontrados foi similar ao das versões anteriores, sem a aplicação da abordagem. Percebe-se que mesmo que não exista conceito de "Zero defeitos", ainda foram identificados erros na aplicação.
- No critério redução de esforço, permitiu a identificação do mesmo número de defeitos que versões anteriores sem aplicação da abordagem, porém o tempo gasto para concluir os testes e o número de recursos utilizados foi menor.

- No critério negócio, houve uma pequena participação para definição de quais funcionalidades seriam mais importantes devido à falta de tempo para se testar tudo, entretanto este foi um dos maiores problemas relatados por Amland, pois a falta de conhecimento de algumas funcionalidades dificultou a análise e produziu resultados não confiáveis.
- No critério processo de gestão de risco, é utilizado uma associação das atividades de gestão de riscos as atividades desta abordagem conforme apresentado nas seções 2.5.1 conforme Figura 2.15 .
- No critério tipos de teste, o autor, não sugere utilizar os tipos de teste de acordo com os riscos identificados.

**Abordagem baseada em Código-fonte Orientado a Objetos:** O autor Rosenberg [24], apresenta uma abordagem e exemplo de aplicação dessa. Neste sentido, a análise desta abordagem foi realizada com base neste estudo e na dissertação de mestrado de Souza [75]

- No critério simplicidade, demonstra ser de simples aplicação com apoio de um ferramental, entretanto caso seja feito manualmente, se torna uma atividade extremamente complexa.
- No critério abrangência, apresenta não estar adaptável, visto que seu foco está somente para o código fonte e classes desenvolvidas. Não existe uma relação das classes com as respectivas funcionalidades. Nesse ponto o foco é apenas na inspeção.
- No critério efetividade, não estar adaptável, visto que seu foco é apenas em código fonte não prevendo nenhum tipo de teste funcional, conforme apresentado na seção 2.3. Apesar de todo o código ser analisado, isso não é suficiente para identificação de erros e garantia da qualidade. A técnica demonstra ser útil, mas não efetiva devido sua abrangência.
- No critério redução de esforço, apresenta mais rapidez em sua aplicação que as demais, entretanto devida sua abrangência e efetividade, não abarcar o contexto completo. Isso justifica o fato de ser mais rápida, considerando ainda o fato de ser realizada com o apoio de ferramental.
- No critério negócio, a abordagem não tem nenhuma cobertura, visto que o foco é o código fonte.
- No critério processo de gestão de risco, a abordagem não tem nenhuma cobertura, visto que o foco é o código fonte.

- No critério tipos de teste, prevê a utilização apenas de inspeção, não abrangendo as demais técnicas. Nesta abordagem o foco aqui é na verificação do código fonte apenas.

**Abordagem Teste de Regressão:** Chen, em sua dissertação de mestrado e em seu artigo,[26, 85] realizou um estudo de caso em um sistema da indústria. Neste sentido, a análise desta abordagem foi realizada com base nestes e em outros estudos dispostos na literatura [75, 100].

- No critério simplicidade, a análise realizada através de questionários submetidos aos desenvolvedores, com questões que os mesmos dominam, não constituiu uma tarefa complexa.
- No critério abrangência, a autora não realiza a atividade de identificação de risco, visto que assume que os casos de testes estão prontos e apresenta também a necessidade de atualização da documentação. Neste sentido a abordagem tem um bom atendimento ao critério, podendo haver alguma restrição, porém de baixo impacto..
- No critério efetividade, segundo a autora a abordagem se mostrou bastante eficiente de acordo com os resultados apresentados. Com a utilização da abordagem foram identificados 9 defeitos e no método tradicional, foi identificado apenas 7, mostrando ser mais efetivo na identificação de defeitos e a cobertura de teste foi maior.
- No critério redução do esforço, segundo a autora, com a ajuda de planilhas automatizadas, todo o processo foi realizado de forma rápida.
- No critério processo de gestão de risco, é utilizado uma associação das atividades de gestão de riscos as atividades desta abordagem conforme apresentado na seção 2.5.1 e de acordo com a Figura 2.15
- • No critério tipos de teste, são previstos tipos de teste de caixa preta, mas a autora não menciona testes de caixa branca, conforme apresentado na seção 2.3.

**Abordagem baseada em Uso:** Besson, em seu artigo [27] apresentou uma abordagem baseada na utilização do software. Neste sentido, a análise desta abordagem foi realizada com base nos resultados neste estudo e com base no trabalho de [100]:

- No critério simplicidade, demonstra ser de simples aplicação, visto que é realizada uma lista de funcionalidades e a partir desta, é realizada uma pesquisa com usuários finais com intuito de identificar as mais utilizadas.

- No critério abrangência, os casos de testes são projetados, com base nos 20% das funcionalidades mais utilizadas e elegidas pelos usuários. Entretanto isso não garante que as funcionalidades não elencadas estejam menos propensas a apresentar falhas.
- No critério efetividade, seguindo a teoria de Pareto apresentada pelo autor, apenas vinte por cento das funcionalidades serão testadas, ou seja, 80% das funcionalidades não são cobertas nos testes.
- No critério redução do esforço, a abordagem é bem aplicável, visto que são testados apenas 20% que foram priorizados junto ao gestor e que são mais utilizados. Entretanto isso impacta os critérios de efetividade e abrangência.
- No critério negócio, a abordagem se mostra aplicável, visto que é realizado uma pesquisa junto aos usuários finais e gestores, com intuito de buscar quais funcionalidades são mais utilizadas através da lista de todas as funcionalidades por meio de uma entrevista.
- No critério processo de gestão de risco, a abordagem não utiliza o processo de gestão de risco associado ao processo de teste.
- No critério tipos de teste, a abordagem não deixa explícito a utilização de tipos de testes, apenas informa que realiza a execução dos casos de testes projetados, para os 20% que foram priorizados.

**Prisma:** Trata-se de um método e ferramenta prática de Teste Baseado em Risco. Erick, em seu artigo [28], apresentou um questionário aplicado com mais de 20 empresas que já haviam aplicado o método PRISMA, visto não ser possível realizar uma comparação entre testes projetos usando o método PRISMA. Neste sentido, considerando os critérios elencados:

- No critério simplicidade, demonstra não ser muito simples, sendo que na pesquisa de facilidade de uso, apresentou resultados de 6.4 em sua média final.

  No critério abrangência, os casos de testes são projetados com base no resultado da matriz de riscos. O que mais tiver risco deve ser mais testado e o que for elencando com menos risco, deve ser menos testado e os tipos de testes a serem aplicados, são subjetivos, dependendo da experiência do gerente de teste ou analista de teste em sua definição.
- No critério efetividade, segundo a pesquisa realizada atende aos critérios com a pontuação de 7.2, entretanto não em sua completude, uma vez que não foi possível medir a eficácia através de resultados reais.

- No critério redução de esforço, a abordagem é bem aplicável, visto apresentar uma matriz de priorização, ao qual o que tem mais risco deve ser testado primeiro e o que tem menos risco deve ser testado posteriormente.
- No critério negócio, o método ser mostra se bem aplicável uma vez que as áreas de negócios participam efetivamente do processo de levantamento de riscos. No critério processo de gestão de risco, a abordagem utiliza como referência a *RBT - Risk Basead Test* entretanto não a ISO31000.
- No critério tipos de teste, a ferramenta deixa explícito que a estratégia de teste, será baseado na matriz de risco a escolha dos participantes do processo, e não existe nenhum método automático para inferir quais tipos de teste fazer para diminuir a probabilidade e impacto de ocorrência de erros em ambiente produtivo.

Com base nos resultados apresentados, percebe-se que nenhuma das abordagens atende a todos os critérios elencados em sua completude. No critério simplicidade praticamente todas as abordagens atendem aos critérios, com exceção do método PRISMA, sendo cada uma em seu contexto específico. No critério abrangência nenhuma delas atendeu em sua completude, ficando os mais próximos a abordagem baseada em Heurística e Teste de Regressão com bom atendimento aos critérios, podendo haver alguma restrição, porém de baixo impacto. Quanto ao critério efetividade, nenhuma das abordagens atendeu em sua completude, o que mostra que nenhuma delas é tão efetiva para a melhoria da qualidade de software, através de testes baseado em riscos. O critério de redução do esforço é contemplado por todas as abordagens, mostrando que para este contexto as abordagens mostram ser bastante efetiva. No critério negócio, apenas três abordagens (Prisma, Regressão e em uso) estão adequadas, mostrando que para a maioria das abordagens o conhecimento negocial não é levado em consideração, permitindo assim falsos positivos na avaliação de risco realizada. Quanto ao critério do processo de gestão de risco, 3 das abordagens (heurística, métricas e regressão) atendem em sua completude, permitindo a identificação, estratégia, análise, mitigação, comunicação e previsão dos riscos. No critério tipos de teste, nenhuma das abordagens permite uma estratégia de avaliação do risco e associação do tipo de teste adequado aos riscos elencados.

No geral, a abordagem de teste de regressão, recebeu uma maior pontuação, entretanto seu foco está mais voltado a testar os impactos das mudanças do que no teste tradicional. Dentre as abordagens apresentadas, apenas uma apresentou uma ferramenta automatizada para apoio na aplicação do processo, entretanto está muito voltada, como as demais, na priorização dos esforços de testes, além de ter deixado a desejar no quesito simplicidade. Nenhuma das abordagens deixa explícito em qual fase do ciclo de vida de desenvolvimento do software a análise de risco é aplicada.

Para uma representação visual de cada abordagem aos critérios elencados, foi utilizado gráfico de radar conforme apresentado na Figura 4.2, para demonstração gráfica da amplitude.

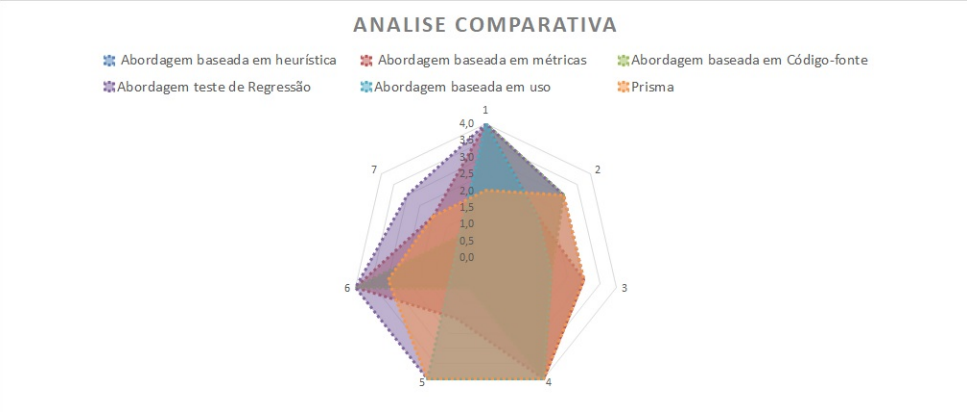



Figura 4.2: Gráfico da Análise Comparativa
Fonte: Elaborado pela Autora

De posse da análise, foi possível identificar alguns *gaps*, abordagens e ferramentas já existente, sendo elas:

- Foco é muito forte na diminuição dos esforços de teste, do que na qualidade;
- Não leva em consideração o conhecimento do negócio em sua totalidade;
- Nem todas as abordagens consideram o processo de gestão de risco;
- Não apresenta uma estratégia de teste adequada para reduzir os riscos da má qualidade de software;
- Não trabalha com a identificação dos riscos;

Considerando os pontos acima expostos, foi possível ter uma visão da composição da nova ferramenta, que incorporou em seu escopo, fácil utilização e aplicação dentro do processo de gestão de risco e teste de software, utilizando como base os critérios de qualidade dispostas na literatura, tal como ISO2510 [2] e Risk Based Test (RBT) [21]. Foi considerado também a opinião das partes interessadas do projeto de software, que envolve, equipe de desenvolvimento, teste, requisitos, gerentes, gestores e usuários finais com intuito de ser mais efetivo na análise de risco.

Considerou-se também a identificação de riscos que afetam as características de qualidade, com vistas a conceder mais pesos a esses critérios. Neste sentido, a ferramenta permitirá também, por meio de quadrantes a seleção das funcionalidades mais e menos críticas, permitindo assim a redução dos esforços de teste através de priorização, mais tendo

como foco sua abrangência e efetividade apresentando de maneira automática, quais tipos de testes realizar aos itens de riscos (Funcionalidades/Requisitos), com intuito a identificar a máxima quantidade de defeitos e assim contribuir para o foco da pesquisa, que é a melhoria da qualidade do produto de software. Alguns pontos fortes identificados na análise apresentada também serão incorporados a nova ferramenta, conforme apresentado na próxima seção, ao qual serão expostos os requisitos e documentação da ferramenta.

## 4.2 Fases de Desenvolvimento da Ferramenta

De posse do conhecimento teórico e da análise comparativa apresentadas na seção 4.1, esta seção apresenta o desenvolvimento da ferramenta de Gestão de Risco para Qualidade de Software, e detalha as tecnologias empregadas, os requisitos implementados, sua arquitetura e estrutura.

### 4.2.1 Concepção

Conforme detalhado no capítulo 1, a concepção da ferramenta proposta surgiu de uma necessidade da empresa RSI apresentar um processo que garanta a entrega de um software de qualidade. E a partir disso, a pesquisadora se fundamentou em técnicas de gestão de riscos que pudessem ser aplicadas ao processo de desenvolvimento de software, a fim de direcionar os testes adequados às funcionalidades presentes, visando assegurar o atendimento às características de qualidade.

### 4.2.2 Requisitos

Para a construção da ferramenta de Gestão de Risco para Qualidade de Software, a elicitação dos requisitos, foi realizada através de um *Benchmarking* e Entrevistas não estruturadas além da revisão da literatura, para a identificação das funcionalidades que irão compor a nova ferramenta, conforme detalhamento a seguir.

#### *Benchmarking*

O levantamento de requisitos, iniciou por meio da RSI Informática, empresa especializada em Qualidade de Software que presta serviço a vários clientes na esfera privada e federal. Neste sentido, foram identificados os clientes da empresa, que utilizam algum tipo de ferramenta de análise de risco voltadas a teste de Software. Foram identificados 5 clientes que aplicam essa prática, conforme detahamento a seguir. Os clientes serão titulados por números, com vistas a preservar a identidade e sigilo de informações.

**Clientes 1, 2 e 3**: Utilizam uma ferramenta nomeada por Mapa de Cobertura de Risco. Esta ferramenta apoia na priorização e definição da estratégia de teste a ser realizada junto aos sistemas, conforme processo de Identificação de Risco, Análise de Risco, Matriz e Risco e Relacionamento e Impacto, conforme descrito a seguir:

**1. Identificação do Risco:**

- Efetua um Brainstorming para listar os riscos potenciais.

- Separa os riscos em: Fatores de Desenvolvimento por meio da probabilidade e Fatores de Negócio por meio do impacto.

- Os riscos, são as funcionalidades dos sistemas, podendo ser caso de uso ou histórias do usuário.

**2. Análise do Risco:**

- Define a lista de Funcionalidades/Requisitos a serem avaliados

- Atribui peso aos Fatores de Risco

- Atribui pontuação considerando a equipe do projeto do ponto de vista dos Fatores de Desenvolvimento (Probabilidade) e Usuários da área de negócio para Fatores de Negócio (Impacto).

- Os fatores de risco, podem ser elencados considerado o negócio, tendo como default os seguintes itens: - Fatores de desenvolvimento: Funcionalidade, Disponibilidade, Usabilidade, Desempenho, Manutenibilidade e Portabilidade - Fatores de Negócio: Efetividade, Produtividade, Segurança Satisfação, Perda Financeira e Danos a Imagem - É acionado uma opção "Gerar Matriz de Risco"

**3. Matriz de Risco**

É apresentada uma matriz de risco bidimensional com quatro quadrantes, sendo:

Quadrante I – Baixa Probabilidade e Baixo Impacto

Quadrante II – Alta Probabilidade e Baixo Impacto

Quadrante III - Baixa probabilidade e Alto impacto

Quadrante IV - Alta probabilidade e Alto impacto

– As funcionalidades de requisitos, são distribuídas dentro dos quadrantes considerando a pontuação que foi realizada na Análise de Risco.

– O Mapa de risco solicita indicar qual estratégia de teste seguirá a depender do quadrante que foi elencando a funcionalidade.

**4. Relacionamento e Impacto**

É apresentado uma matriz de rastreabilidade relacionando as funcionalidades do sistema em dois eixos, para que seja realizado de maneira manual o relacionamento entre as funcionalidades para uma avaliação de impacto a mudança informando: 0 – Quando não há relacionamento, 1 – Quando existe um relacionamento indireto e 2 – Quando há um relacionamento direto.

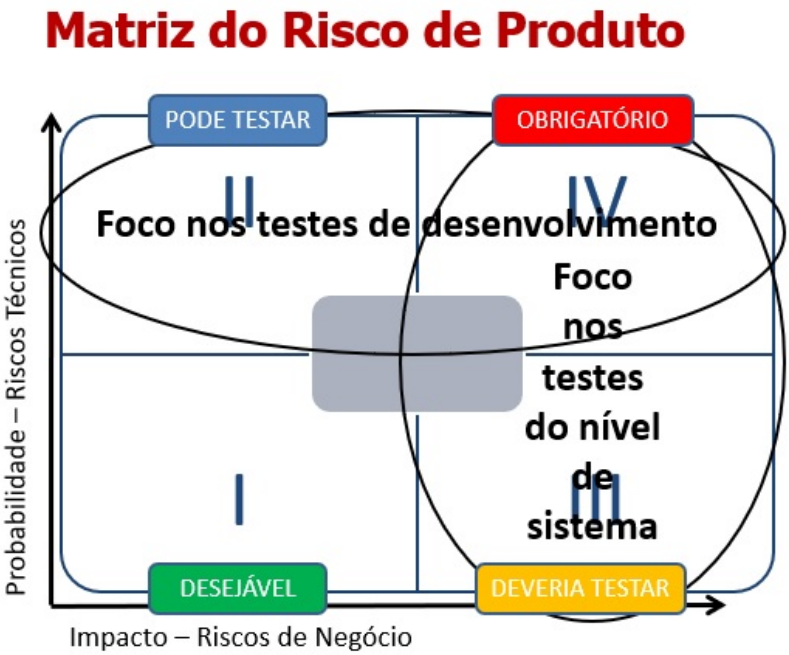



Figura 4.3: Matriz do Mapa de Cobertura de Risco
Fonte: Cliente da RSI Informática

**Cliente 4 e 5**: Utilizam uma ferramenta nomeada por Análise de Riscos no Modelo Ágil. Esta ferramenta apoia na priorização manual em quais tipos de testes e quais casos de testes realizar, através de uma matriz de Risco, conforme apresentado na Figura 4.4.

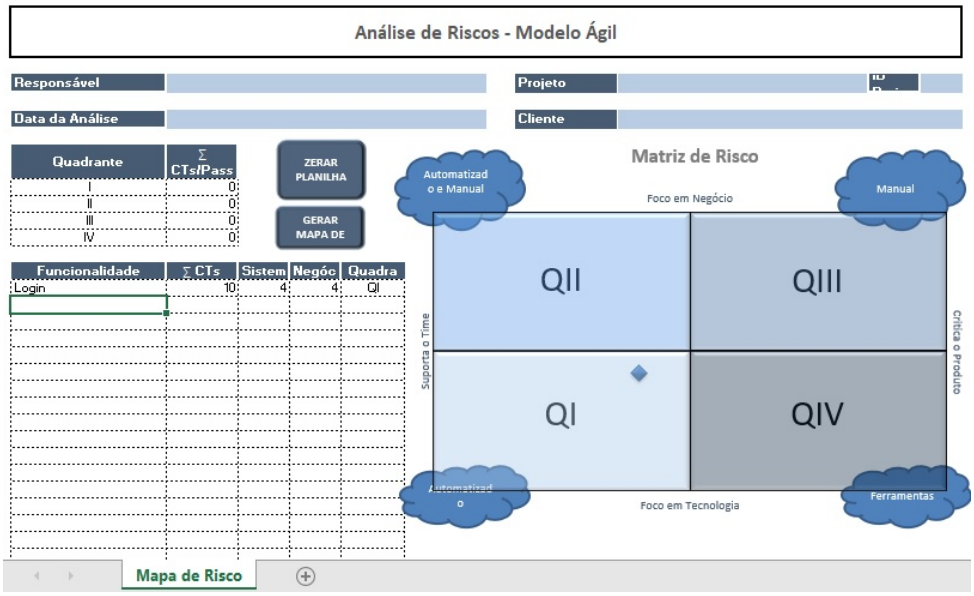



Figura 4.4: Análise de Risco - Modelo Ágil
Fonte: Cliente da RSI Informática

Esta ferramenta é utilizada para softwares desenvolvidos na metodologia ágil, conforme apresentado na seção 2.1.5 do referencial teórico. Sua definição é totalmente manual não tendo nenhum método de Gestão de Risco e Teste de software Baseado em Risco atrelado a sua composição. Ela permite uma priorização de atuação dentre os quadrantes elencados, sendo que as funcionalidades que foram indicadas no quadrante I, devem ser realizados testes automatizados, as funcionalidades indicadas no quadrante II deverão ser realizadas testes manuais e automatizados, as indicadas quadrante III somente realização de testes manuais e as dos quadrantes IV testes utilizando ferramental.

Ao final do *Benchmarking*, foram realizadas pesquisas não estruturadas, junto a equipe que realiza a aplicação do método e ferramenta junto aos clientes, para buscar uma percepção sobre sua utilização.

Para os clientes 1, 2 e 3 foi sinalizado que, a aplicação da ferramenta apoia muito na priorização dos testes, indicando quais funcionalidades são críticas e quais não são. Apoia também na definição de quais tipos de testes realizar a depender do quadrante em que ela ficou localizada, permitindo de uma forma subjetiva indicar quais testes devem ser realizados para cada quadrante. Entretanto, sinalizam que é complexo o entendimento do método, principalmente pelas áreas de negócio na fase de análise de risco, onde os participantes precisam pontuar qual a probabilidade de ocorrência e impacto, caso ocorra, junto às características de qualidade.

Para os clientes 4 e 5, foi sinalizado pela equipe que utiliza a ferramenta, que ela é de simples aplicação, no entanto é totalmente manual e subjetiva, uma vez que é declarado

qual quadrante a funcionalidade será posicionada, dependendo muito da experiência do Analista ou Gerente de Teste para uma boa análise.

Uma vez concluído o *Benchmarking*, foi realizada uma entrevista não estruturada junto a especialistas da empresa RSI Informática, com conhecimento nas ferramentas internas de gestão de risco em teste de Software, com vistas a buscar melhorias que irão compor a nova ferramenta, conforme apresentado a seguir.

### Entrevista não estruturada

A pesquisa não estruturada foi realizada junto a 8 especialistas da empresa RSI Informática dentre um corpo técnico de 78 profissionais, com este mesmo perfil, representando 10% do corpo técnico mais especializado da empresa e com conhecimento nas ferramentas internas de gestão de risco voltadas a teste de software. Teve como intuito buscar as necessidades a serem incorporadas à nova ferramenta. Neste sentido, a identidade dos especialistas será preservada e elecandas por numeração. Na oportunidade foi realizada a seguinte pergunta aos entrevistados:

- O que você acha que agregaria valor à nova ferramenta de gestão de risco voltada à qualidade de software?

A Tabela 4.2 apresenta o resultado da pesquisa.

Tabela 4.2: Entrevista não estruturada com especialistas

| Especialista | Resposta |
|---|---|
| **Especialista 1** | - A nova ferramenta deve ser de fácil entendimento às áreas de negócios para sua efetiva participação no processo de gestão de risco.<br>- Ferramenta web para aplicação do processo. |
| **Especialista 2** | - Uma área para tratar os riscos de produto, seria bem interessante para a nova ferramenta, uma vez que todas as ferramentas tratam apenas os itens de riscos, ou seja, os requisitos. |
| **Especialista 3** | - Uma ferramenta automatizada com vistas a manter uma base de conhecimento a ser utilizada em projetos de testes futuros. |
| **Especialista 4** | - Deixar mais simples a forma de apresentar as características de qualidade para pontuação da equipe do projeto e da área de negócio. |
| **Especialista 5** | - Seria um grande diferencial ter uma ferramenta que indique além da priorização qual tipo de teste realizar para cada funcionalidade.<br>Ou seja, uma estratégia de teste a ser aplicada para mitigar os riscos da má qualidade. |
| **Especialista 6** | -Associar a análise de risco com indicadores de execução de teste. |
| **Especialista 7** | - Ter a ferramenta de forma web e automatizada para manter uma base de conhecimento. |
| **Especialista 8** | - Ser mais explicativa para apoio na aplicação do processo.<br>- Inclusão dentro do processo de teste e de desenvolvimento de software. |

Considerando os pontos expostos pelos os especialistas, foi possível compor um protótipo para a nova ferramenta, que teve como base inicial o Mapa de Cobertura de Risco, considerando que ele já abrange os conceitos de teste de software baseado em risco e atributos das abordagens apresentadas no referencial teórico.

### Documentação da ferramenta de Gestão de Risco para Qualidade de Software

A construção da ferramenta de Gestão de Risco para Qualidade de software, foi estruturada em 4 módulos, conforme apresentado na Figura 4.5, sendo o primeiro módulo de administrador, que abrange o controle de acesso da ferramenta, o segundo módulo de cadastro, que compreende todos os cadastros estruturantes da ferramenta, o terceiro módulo de Gestão de Risco, que permite a realização da Gestão de Risco para Qualidade de Software e o módulo de indicadores que permite o monitoramento da gestão de testes para acompanhamento da estratégia de risco indicada pela ferramenta. Esta seção, apresenta todas as regras e requisitos para a construção da ferramenta.


Figura 4.5: Módulos da Ferramenta de Gestão de Risco para Qualidade de Software
Fonte: Elaborado pela autora

Considerando os módulos estruturados, foi construído um diagrama de caso de uso,

conforme Figura 4.6 para representação das principais funcionalidades e atores da ferramenta e suas devidas iterações.

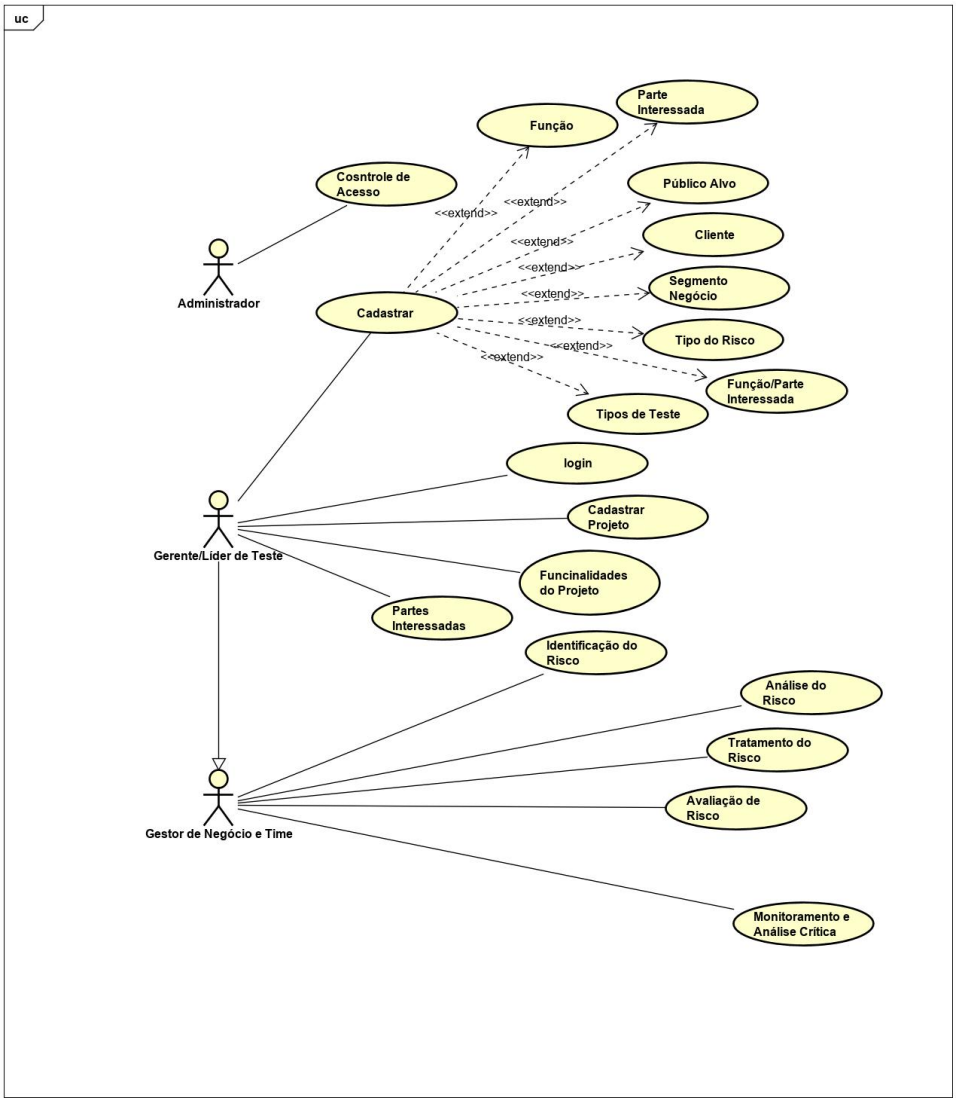



Figura 4.6: Diagrama de Caso de Uso
Fonte: Elaborado pela autora

Neste cenário, a elipse é representada pelas funcionalidades, os bonecos são representados pelos atores que irão interagir com a ferramenta e com as funcionalidades, e as setas de conexões, representam as iterações e heranças entre casos de uso e atores. O *extend* representa o relacionamento de inclusão opcional, o que indica que o ponto de extensão que pode ou não ocorrer, a depender da necessidade. São representados todos os atores e funcionalidades da ferramenta. A seguir, são descritas suas respectivas funcionalidades implementados nesta primeira versão da ferramenta de Gestão de Risco para Qualidade de Software, agrupadas por módulo. Todas são compostas pelo protótipo da tela e suas respectivas funcionalidades e regras implementadas, na construção da ferramenta. **Módulo Administrador**

Este módulo abrange as funcionalidades de controle de acesso e login, que permitem concessão e acesso aos usuários na ferramenta de Gestão de Risco para Qualidade de software, conforme demostrando na Figura 4.7. **Controle de Acesso**: Permite a pesquisa, inclusão, alteração e exclusão de usuários e níveis de acesso a ferramenta.

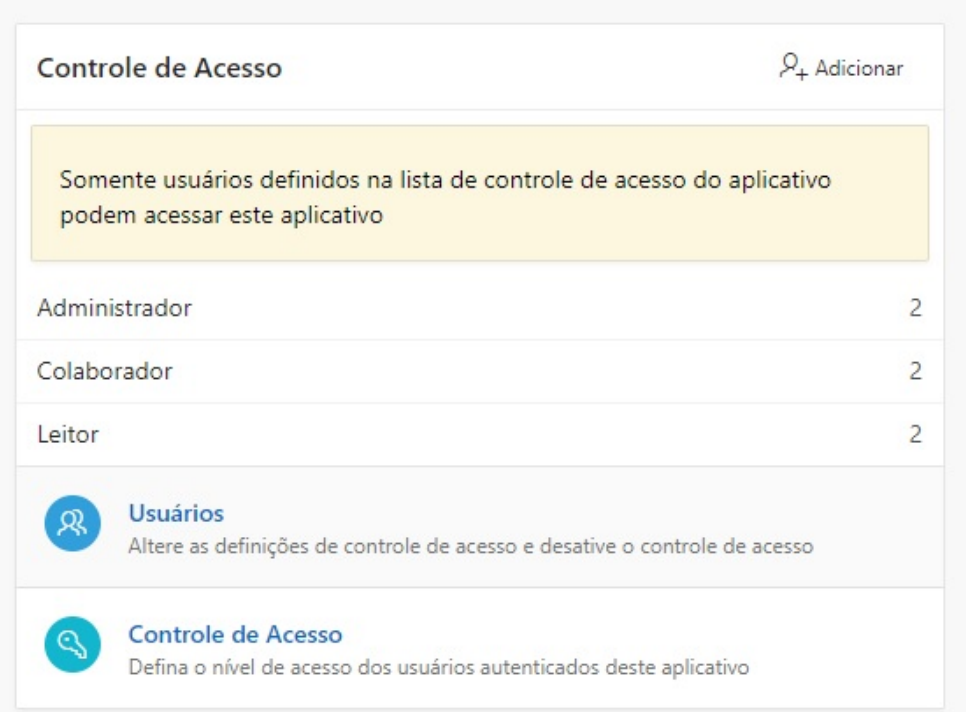



Figura 4.7: Tela de controle de acesso
Fonte: Elaborado pela autora

**Login:** Permite que usuários cadastrados tenham acesso às demais funcionalidades do sistema que exijam autenticação. Esta tela é composta por um login e senha, conforme apresentada na Figura 4.8.


Figura 4.8: Tela de Login
Fonte: Elaborado pela autora

1. O cadastro inicial dos usuários é realizado pelo administrador da ferramenta, que realiza o cadastro de login e senha inicial que pode ser alterada pelo usuário no primeiro acesso da ferramenta;
2. O acesso à ferramenta é permitido somente a usuários devidamente cadastrados;
3. O login é dado por uma usuário e senha;

### Módulo de Cadastro

Este módulo, abrange todas as funcionalidades que são estruturantes da ferramenta, permitindo o cadastro das informações a serem utilizadas no módulo de Gestão de Risco, conforme apresentado a seguir. Este acesso é realizado pelo administrador da ferramenta.

- **Página Inicial:** Tela inicial da ferramenta, conforme apresentado na Figura 4.9.

  No canto superior diretiro da ferramenta, é posível fazer o *download* do manual de instruções de utilização da ferramenta de Gestão de Risco para Qualidade de Software, conforme apresentado no Apêncide A.

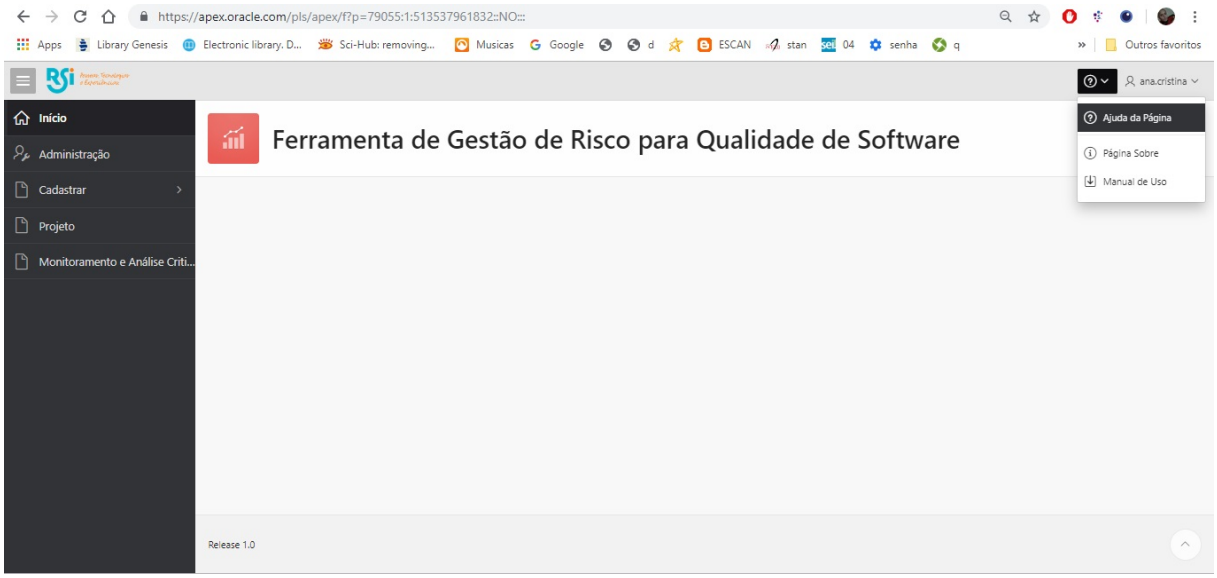



Figura 4.9: Tela Inicial
Fonte: Elaborada pela autora

O Módulo de Gestão de Risco da ferramenta utiliza como base a Matriz de Probabilidade e Impacto para classificação dos Riscos, conforme Figura 4.10.

| MATRIZ DE DEFINIÇÃO DE PROBABILIDADE | | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| | **Nulo** | **Muito baixa** | **Baixa** | **Média** | **Alta** | **Muito Alta** |
| **Probabilidade** | < 1, probabilidade minima de ocorrer | De 1 - 10% de probabilidade de ocorrer | De 11% à 30% de probabilidade de ocorrer | De 31% à 50% de probabilidade de ocorrer | De 51% à 70% de chance de ocorrer | Mais > 70% de chance de ocorrer |

| MATRIZ DE DEFINIÇÃO DE IMPACTO | | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| | **Nulo** | **Muito baixa** | **Baixa** | **Média** | **Alta** | **Muito Alta** |
| **Impacto** | Nenhum impacto sobre a funcionalidade | Impacto secundário sobre funções secundárias. | Impacto secundário sobre a funcionalidade geral. | Algum impacto em área funcionais essenciais. | Impacto significativo sobre a funcionalidade geral. | Impacto muito significativo sobre a funcionalidade geral. |

Figura 4.10: Matriz de definição de probabilidade e impacto
Fonte: Adaptado pela Autora[80]

A Matriz foi definida conforme conceitos extraídos do referencial teórico apresentado na seção 2.5.1 deste trabalho. Será utilizada para a análise qualitativa da probabilidade e impacto, em que se deve questionar qual a probabilidade de ocorrência de falha e qual o impacto, caso ocorra, representadas na ferramenta por meio das funcionalidades de "Identificação de Risco “e "Análise do Risco “ que abrangem o módulo de Gestão de Risco. Para o processo de gestão de Qualidade para software, essa funcionalidade está relacionada com a Fase de Comunicação e Consulta, disposta na seção 4.3.1 deste trabalho.

- **Cadastrar Função:** Permite a pesquisa, criação, alteração e exclusão da função das partes interessadas que irão participar do processo de gestão de risco para Qualidade de software. A tela é apresentada na Figura 4.11 que traz consigo alguns exemplos de funções já registradas.

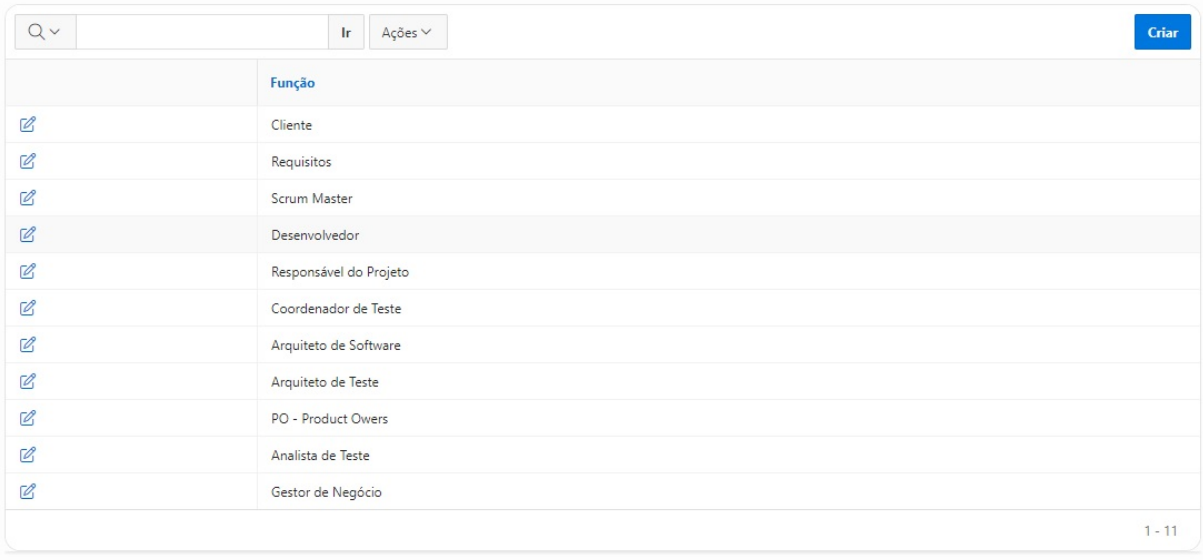



Figura 4.11: Tela de Cadastrar Função
Fonte: Elaborado pela autora

Essas funções representam o papel das partes interessadas desempenhadas dentro do projeto do produto a ser avaliado. Neste sentido, para o processo de gestão de risco para Qualidade de software, essa funcionalidade abrange as Fases de Comunicação e Consulta e Planejamento, disposta nas seções 4.3.1 e 4.3.2 deste trabalho.

- **Cadastrar Partes Interessadas:** Permite a pesquisa, criação, alteração e exclusão das partes interessadas, que irão participar do processo de gestão de risco para Qualidade de software. A tela é apresentada na Figura 4.12 que traz consigo alguns exemplos de partes interessadas já cadastrados.

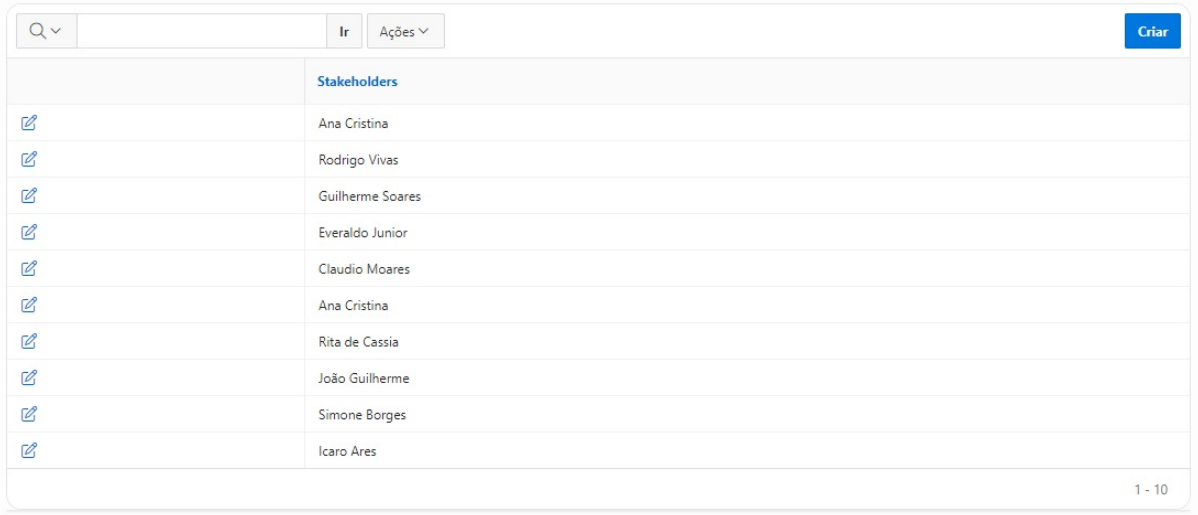



Figura 4.12: Tela de Cadastrar Stakeholders
Fonte: Elaborado pela autora

São mapeados nesta funcionalidade, todas as partes interessadas que irão participar do processo de Gestão de Risco para Qualidade de software que abrange neste contexto as Fases de Comunicação e Consulta e Planejamento, disposta nas seções 4.3.1 e 4.3.2 deste trabalho.

- **Cadastrar Função das Partes Interessadas:** Permite a associação das partes interessadas com a função. O sistema lista todas as funções já cadastradas através dos casos de uso "Cadastrar função" e "Cadastrar Partes Interessadas", conforme apresentado na Figura 4.13.

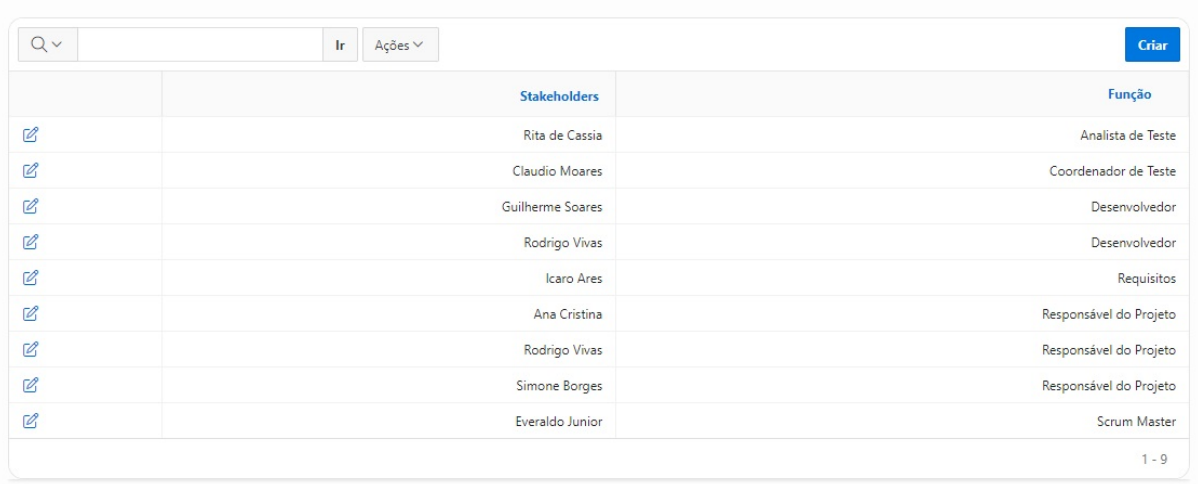



Figura 4.13: Tela de Cadastrar Função das Partes Interessadas
Fonte: Elaborado pela autora

São listados nessa funcionalidade, as partes interessadas e a função para o processo de aplicação da ferramenta de Gestão de Risco para Qualidade de software. Essa funcionalidade abrange as Fases de Comunicação e Consulta e Planejamento, disposta nas seções 4.3.1 e 4.3.2 deste trabalho.

- **Cadastrar Cliente:** Permite a pesquisa, criação, alteração e exclusão dos clientes/organização que irão participar do processo de gestão de risco para Qualidade de Software. A tela é apresentada na Figura 4.14 que traz consigo alguns exemplos já cadastrados.

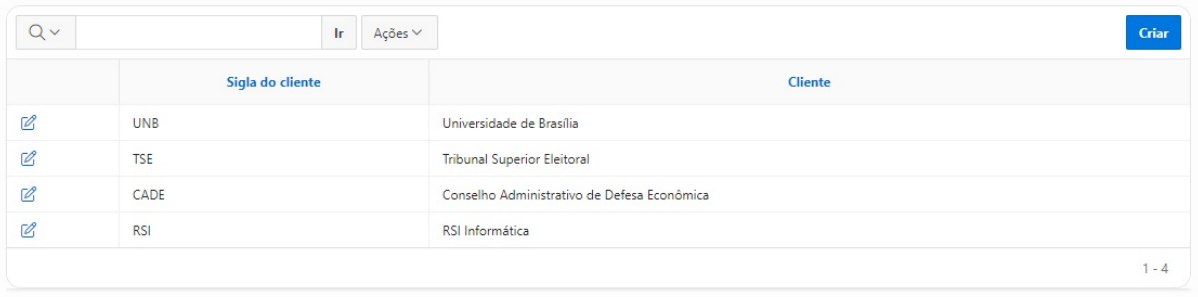



Figura 4.14: Tela de Cadastrar Cliente
Fonte: Elaborado pela autora

São mapeados nessa funcionalidade, todos os clientes que irão aplicar/utilizar a ferramenta de Gestão de Risco para Qualidade de Software. Essa funcionalidade abrange, dentro do processo de gestão de risco para Qualidade de software, as Fases de Comunicação e Consulta e Planejamento, disposta nas seções 4.3.1 e 4.3.2 deste trabalho.

- **Cadastrar Segmento de Negócio:** Permite a pesquisa, criação, alteração e exclusão de segmento de negócio em que o produto de software atua. A tela é apresentada na Figura 4.15 que traz consigo alguns exemplos de segmento de negócio já cadastrados.

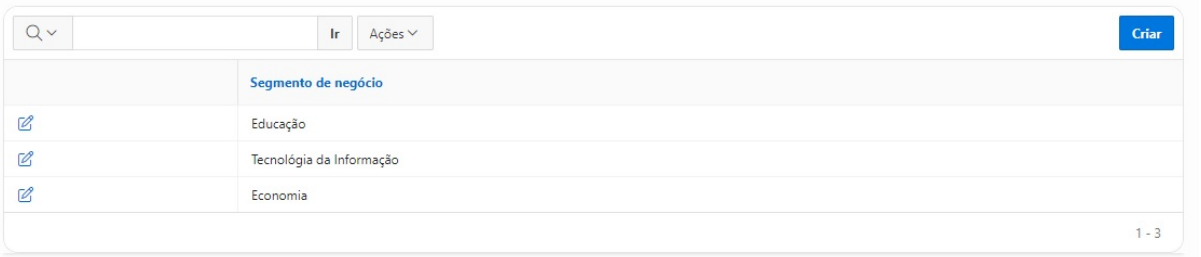



Figura 4.15: Tela de Cadastro do Segmento de Negócio
Fonte: Elaborado pela autora

É mapeada nessa funcionalidade, a área de atuação de negócio em que o cliente atua e que apoiará para base históricas futuras na aplicação da ferramenta de Gestão de Risco para Qualidade de Software. Essa funcionalidade abarca as Fases de Comunicação e Consulta e Planejamento, dispostas nas seções 4.3.1 e 4.3.2 deste trabalho.

- **Cadastrar Tipo de Público Alvo:** Permite a pesquisa, criação, alteração e exclusão de público alvo. A tela é apresentada na Figura 4.17 que traz consigo alguns exemplos de públicos alvo já cadastrados.

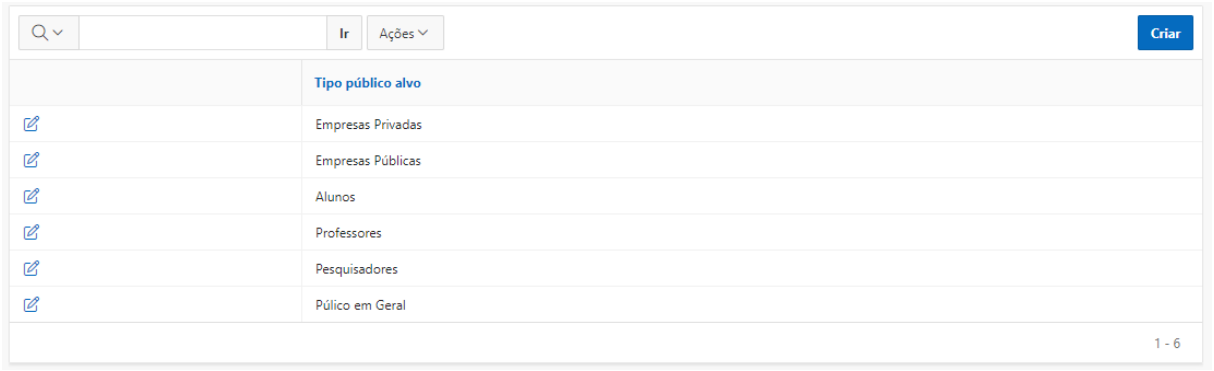



Figura 4.16: Tela de Tipo de Público Alvo
Fonte: Elaborado pela autora

É mapeado nesta funcionalidade, o público alvo que vai utilizar o produto de software a ser avaliado na ferramenta. Considerando o processo de gestão de risco para Qualidade de software, essa funcionalidade engloba as Fases de Comunicação e Consulta e Planejamento, disposta nas seções 4.3.1 e 4.3.2 deste trabalho.

- **Cadastrar Tipo de Risco:** Permite a pesquisa, criação, alteração e exclusão da sigla e tipos de características de qualidade e seu relacionamento com os tipos de fatores de risco. A tela é apresentada na Figura 4.17 que traz consigo um exemplo de Tipo de Risco já cadastrado.


Figura 4.17: Tela de Tipo de Risco
Fonte: Elaborado pela autora

São mantidos nesta funcionalidade todos os atributos de qualidade utilizados para análise do risco do ponto de vista de Fatores de Desenvolvimento (Probabilidade) e Fatores de Negócio (Impacto).A Figura 4.18, apresenta todas as características cadastradas nesta versão da ferramenta e um pequeno resumo de cada uma delas, que serão utilizadas na avaliação do produto de software e são detalhadas no referencial teórico na seção 2.2.2.

| Seleção dos Fatores de Risco - ISO25010 - Caracteristica de Qualidade para Software | |
|---|---|
| **Descrição dos Fatores de Risco - Qualidade Interna e Externa** | |
| **Fatores de Desenvolvimento (Probabilidade)** | |
| Adequação funcional | Refere-se ao grau em que o produto de software fornece a função que atende às necessidades explícitas ou implícitas quando utilizado sob condições específicas. Para o produto de software refere-se aos requisitos funcionais e não funcionais. |
| Confiabilidade | Refere-se à capacidade do software em manter seu nível de operação típico, sob condições estabelecidas, período de tempo específico ou número de operações específico. |
| Usabilidade | Refere-se à capacidade que um software tem de ser entendido, aprendido, utilizado e atraente para o usuário quando utilizado sob determinadas condições, para alcançar eficácia, eficiência e satisfação em um contexto específico de uso |
| Eficiência de desempenho | Refere-se à capacidade que um software tem de fornecer um desempenho apropriado com relação à quantidade de recursos utilizados, sob condições estabelecidas. |
| Manutenibilidade | Refere-se à capacidade de realizar modificações específicas no sistema para corrigir defeitos ou atender a novos requisitos, facilidade de realizar manutenções futuras ou adaptar-se a um ambiente alterado. |
| Portabilidade | Refere-se à habilidade do software para ser transferido de um ambiente para outro, com configuração diferenciada de hardware e/ou software. |
| Compatibilidade | Refere-se a capacidade que um software pode trocar informações com outros produtos, sistemas ou componentes, e/ou executar funções em um mesmo ambiente de hardware ou software. Muito voltado a integração com outras interfaces. |
| Segurança | Refere-se a grau de segurança em que o produto de software protege informações e dados para que pessoas ou outros produtos ou sistemas têm o grau de acesso aos dados adequado aos seus tipos e níveis de autorização. |
| **Descrição dos Fatores de Risco - Qualidade em Uso** | |
| **Fatores de Negócio (Impacto)** | |
| Eficácia | Refere-se à capacidade do produto de software em possibilitar aos usuários atingir metas especificadas com precisão e integridade. |
| Eficiência | Refere-se à capacidade do produto de software em possibilitar aos usuários atingir metas especificadas com completeza, num contexto de uso especificado. |
| Satisfação | Refere-se à capacidade do produto de software em satisfazer usuários, num contexto de uso especificado. |
| Ausência de Risco | Refere-se à capacidade do produto de software mitigar o risco potencial para a situação econômica, vida humana, saúde ou meio ambiente e a imagem. |
| Cobertura de contexto | Refere-se à capacidade do produto de software pode ser utilizado com eficácia, eficiência, isenção de erros e satisfação em contextos específicos de uso e em contextos além daqueles inicialmente explicitamente |

Figura 4.18: Característica de Qualidade
Fonte: Adaptado pela autora [2]

As características apresentadas são demonstradas nas funcionalidades de "Identificação de Risco "e "Análise de Risco" da ferramenta que serão detalhados nesta seção. A Sigla, representa apenas uma redução nas características de qualidade, para facilitar a manutenção da ferramenta. Para o processo de gestão de risco para Qualidade de software, essa funcionalidade envolve as Fases de Comunicação e Consulta e Planejamento, dispostas nas seções 4.3.1 e 4.3.2 deste trabalho.

- **Cadastrar Tipos de Teste:** Permite a pesquisa, criação, alteração e exclusão dos tipos de testes, conforme apresentado na Figura 4.19, que serão utilizados para minimizar os riscos da má qualidade do produto de acordo com os resultados desta ferramenta.

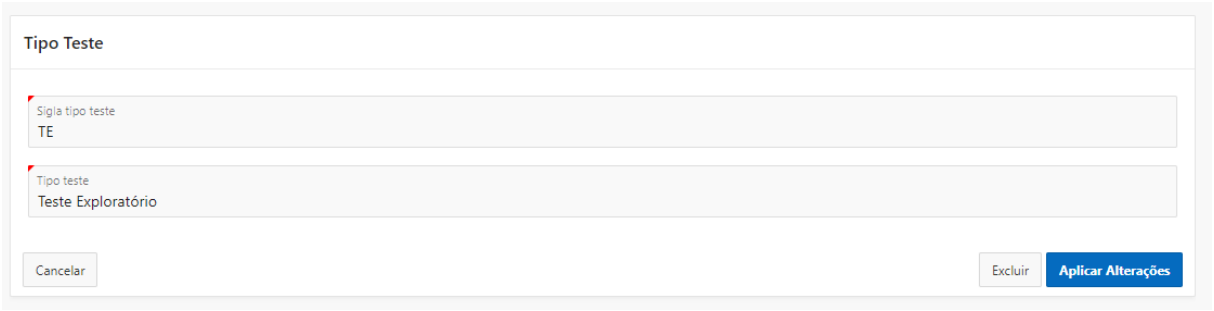



Figura 4.19: Tela de Cadastro de Tipo de Teste
Fonte: Elaborado pela autora

Os tipos de testes cadastrados nesta tela, serão apresentados na funcionalidade "Tratamento de Risco" constante no módulo de Gestão de Risco, com a definição da estratégia de teste para minimizar a má qualidade do Software. No processo de aplicação da ferramenta, essa funcionalidade abrange as Fases de Comunicação e Consulta e Planejamento, dispostas nas seções 4.3.1 e 4.3.2 deste trabalho.

#### Módulo de Gestão de Risco

Esse módulo, abrange todas as funcionalidades que são inerentes à aplicação da Gestão de Risco para Qualidade de Software, conforme listados a seguir:

- **Cadastrar Projeto:** Essa funcionalidade apresenta a pesquisa, alteração, exclusão e cadastro de projetos a serem avaliados por meio da Gestão de Risco para Qualidade do software, conforme apresentado na Figura 4.20.

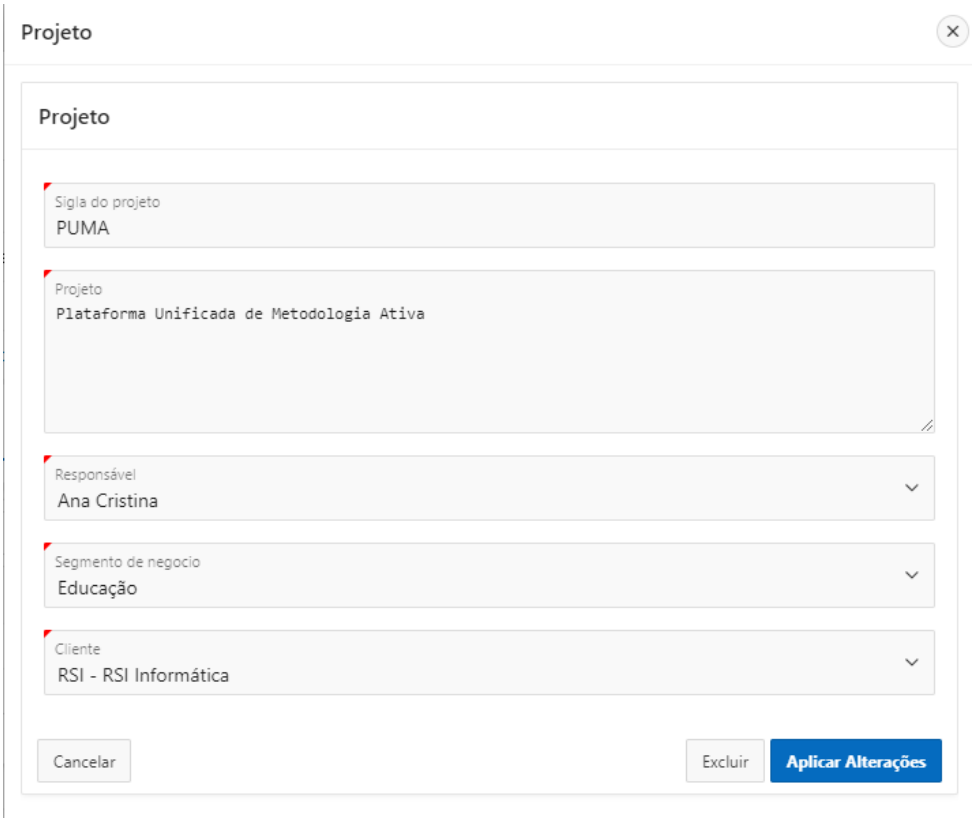



Figura 4.20: Tela de Cadastro de Projeto
Fonte: Elaborado pela autora

São apresentados os seguintes campos para cadastro:

1. Sigla e Nome do projeto, que reflete o projeto em que o produto está sendo desenvolvido, a ser a avaliado pela ferramenta de Gestão de Risco para Qualidade de Software;
2. Responsável, que reflete o principal patrocinador do produto de Software em avaliação;
3. Segmento de Negócio, que reflete o segmento de atuação do produto de Software. É apresentada a listagem para seleção de segmentos já cadastrados por meio da funcionalidade "Cadastrar Segmento de Negócio";
4. Cliente, que reflete a organização que será avaliada pela Gestão de Risco para Qualidade de Software. É apresentada a listagem de clientes já cadastrados, por meio da funcionalidade "Cadastrar Cliente";
5. Público Alvo, que reflete o público que irá acessar o produto de Software. É apresentado a listagem de público alvo já cadastro na ferramenta, por meio da funcionalidade "Cadastrar Tipo de Público Alvo".

Considerando o processo de gestão de risco para Qualidade de software, essa funcionalidade abrange as Fases de Comunicação e Consulta, Planejamento e Registro e Reports, dispostas nas seções 4.3.1, 4.3.2 e 4.3.8 deste trabalho.

- **Partes Interessadas do Projeto:** Essa funcionalidade apresenta a pesquisa, alteração, exclusão e cadastro de partes interessadas, conforme apresentado na Figura 4.21.

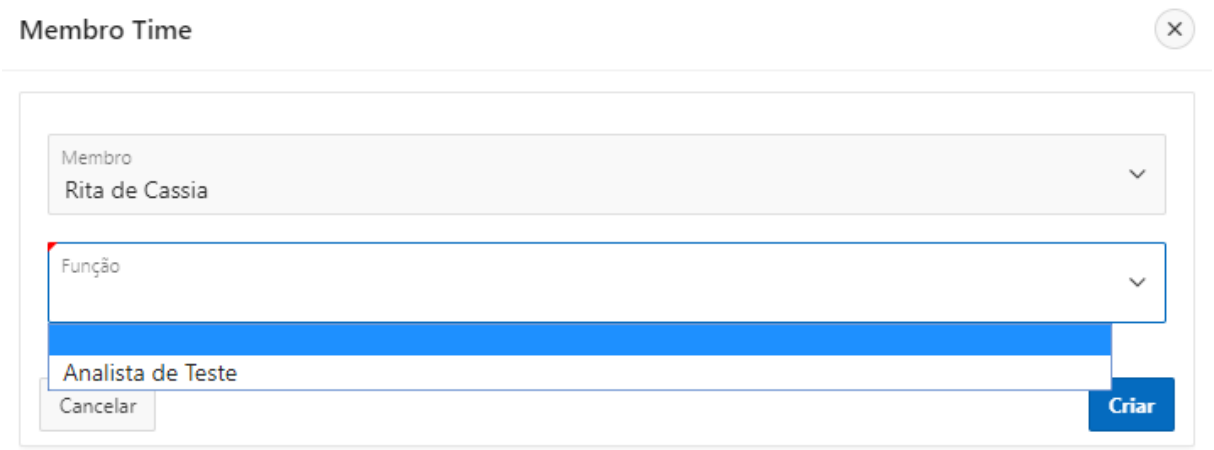



Figura 4.21: Tela de Cadastro de Partes Interessadas do Projeto
Fonte: Elaborado pela autora

Devem ser cadastradas nessa funcionalidade, todas as partes interessadas e suas respectivas funções, que irão participar do Processo de Gestão de Risco para Qualidade de Software. As partes interessadas e função listadas nessa funcionalidade, são provenientes das informações já cadastradas nas funcionalidades de "Cadastrar Partes Interessadas" e "Cadastrar Função" do módulo de cadastro. Para o processo de gestão de risco para qualidade de software, essa funcionalidade abarca as Fases de Comunicação e Consulta, Planejamento e Registro e Reports, disposta nas seções 4.3.1, 4.3.2 e 4.3.8 deste trabalho.

- **Cadastrar Funcionalidades do Produto:** Permite a pesquisa, alteração, exclusão e cadastro das funcionalidades do produto a ser avaliado junto a Gestão de Risco para Qualidade de Software, conforme apresentado na Figura 4.22.

  Devem ser cadastros nessa funcionalidade todos os requisitos que serão avaliados na ferramenta, podendo abranger: Fluxos de Negócio, Casos de Uso, História do Usuário. Esses requisitos, por sua vez, serão apresentados na funcionalidade de "Análise e Risco"para uma avaliação qualitativa baseada nos requisitos do produto de Software. Para o processo de gestão de risco para Qualidade de software, essa

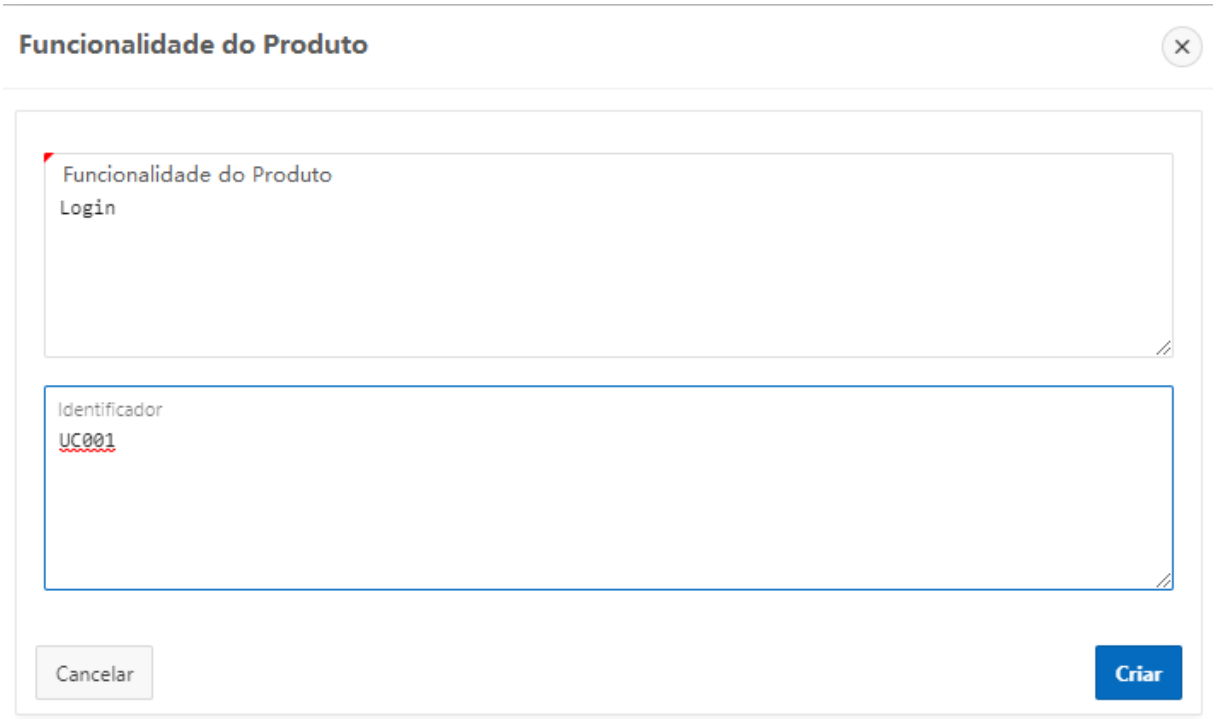



Figura 4.22: Tela de Funcionalidades do Projeto
Fonte: Elaborado pela autora

funcionalidade compreende as Fases de Comunicação e Consulta, Planejamento e Registro e Reports, disposta nas seções 4.3.1, 4.3.2 e 4.3.8 deste trabalho.

- **Identificação do Risco:** Permite a pesquisa, alteração, exclusão e cadastro dos eventos de riscos ao produto de Software, conforme apresentado na Figura 4.23.

  Nessa funcionalidade, devem ser mapeados todos os possíveis eventos de riscos inerentes ao produto de Software. Este processo de identificação de risco do produto, permite aos projetos de desenvolvimento a exposição das incertezas que podem ocorrer com o produto. A seguir, todos os campos que devem ser preenchidos obrigatoriamente nesta tela.

  1. Evento de Risco, deve ser registrado o risco inerente à qualidade do produto de Software.
  2. Característica de Qualidade, deve ser selecionado a partir da listagem apresentada, conforme ISO25010 [2], tendo como referência a seguinte pergunta: Qual característica de qualidade o risco elencado, pode afetar, caso ocorra?
  3. Probabilidade, deve ser pontuado seguindo a escala apresentada na Figura 4.10. Representa a probabilidade de ocorrência do risco.


Figura 4.23: Tela de identificação do Risco
Fonte: Elaborado pela autora

4. Impacto, deve ser pontuado seguindo a escala apresentada na Figura 4.10. Representa o impacto de ocorrência do risco.

Uma vez concluída toda a identificação, a ferramenta irá calcular o peso de cada risco, e irá atribuí-lo às perguntas constantes na funcionalidade de "Análise de Risco"relacionadas às características de qualidade elencadas. O peso é calculado pela multiplicação da probabilidade e impacto, conforme apresentado na fórmula da Equação 4.1:

$$P * I \tag{4.1}$$

Caso tenha-se mapeado mais de 1 risco que tenham a mesma característica de qualidade, é atribuída uma média, conforme apresentado na fórmula da Equação 4.2:

$$\sum(P * I)/QR \tag{4.2}$$

Onde P representa a probabilidade e I o impacto e QR a quantidade de riscos elencados.

Quando não forem mapeados, riscos associados a um critério de qualidade, será atribuído à pergunta peso 1, uma vez que não terá interferência na base de cálculo.

A Figura 4.24 ilustra a atribuição de pesos, conforme apresentado a seguir.

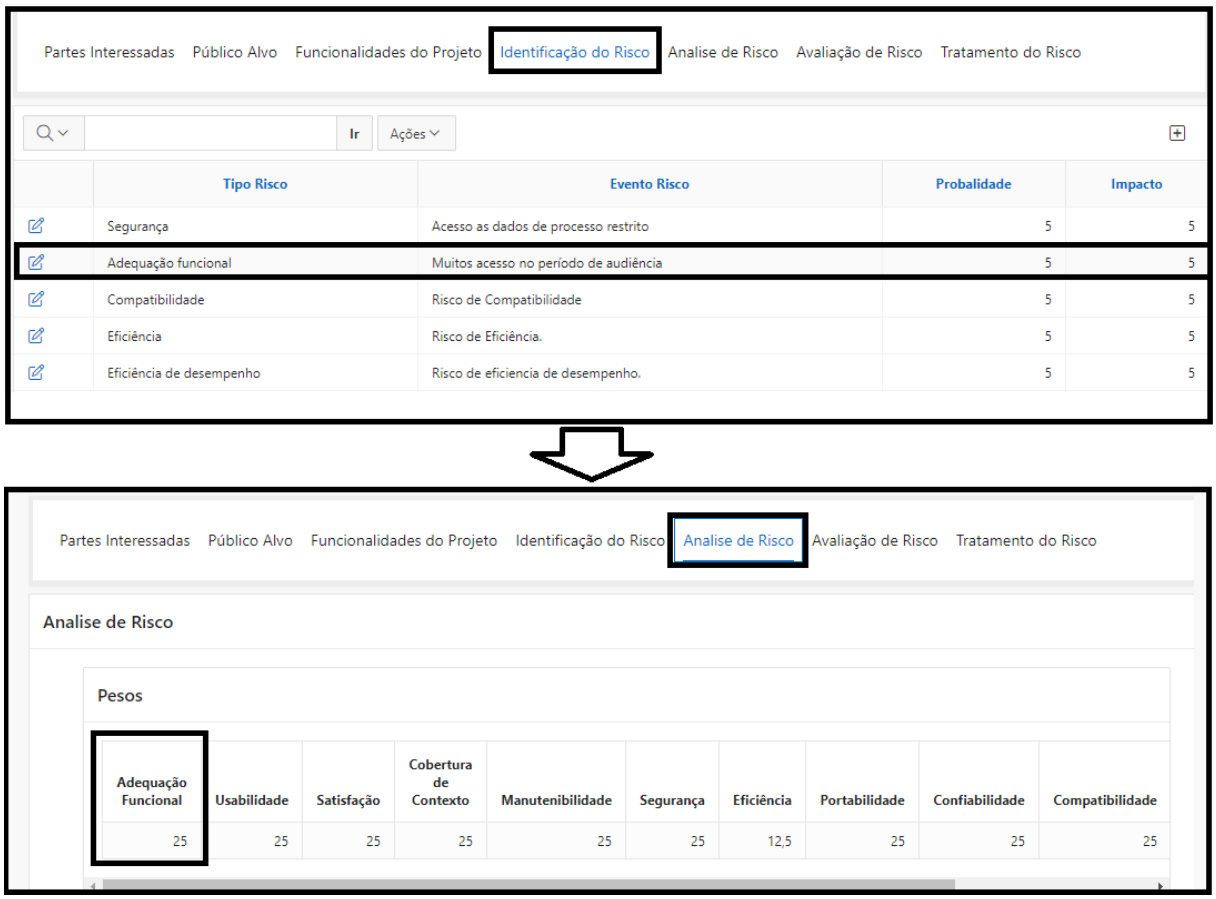



Figura 4.24: Exemplo de Atribuição de Pesos às Características de Qualidade
Fonte: Elaborado pela autora

Neste sentido, as características de qualidade que tem riscos terão possuirão maiores pesos em detrimento das que não tem riscos. A tela de "Análise de Risco" apresenta os pesos registrados nesta funcionalidade. Considerando o processo de gestão de risco para Qualidade de software, este processo é abarcado pela seção 4.3.3, constante na aba de Identificação do Risco.

- **Análise do Risco:** Permite a pesquisa, inclusão, alteração e exclusão da análise de risco, por meio da pontuação das perguntas associadas às características de qualidade e aos itens de riscos (Funcionalidades/Requisitos), conforme apresentado na Figura 4.25.

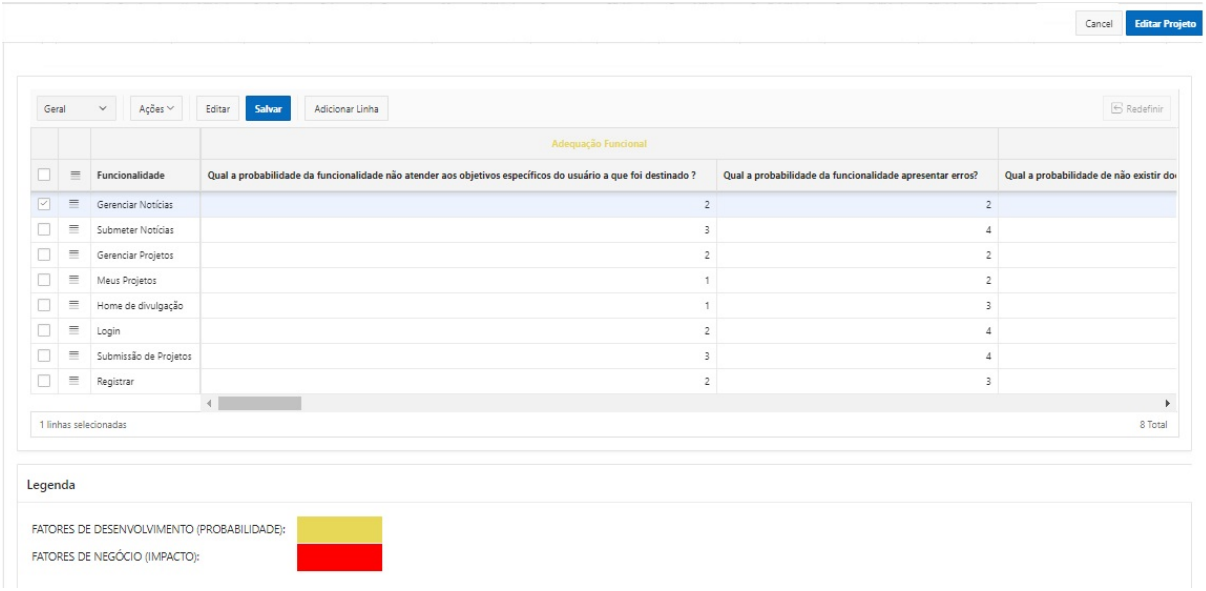



Figura 4.25: Análise dos Riscos para Qualidade de Software
Fonte: Elaborado pela autora

A tela é representada pelos seguintes campos:

1. Pesos, são apresentados sem permissão de edição, referentes às características de qualidade que são proveniente da funcionalidade de Identificação de Risco, conforme apresentado nas seções anterior.
2. Funcionalidades, são listadas, sem permissão de edição, todas as funcionalidades cadastradas na tela "Funcionalidades do Projeto".
3. Ações, permite realizar o download em CSV das perguntas elencadas na funcionalidade e suas devidas classificações.
4. Perguntas, são apresentadas todas as perguntas cadastradas referentes às características de qualidade a serem avaliadas para os itens de riscos, que são representados pelas funcionalidades/requisitos do produto de software a ser avaliado de acordo com a Figura 4.26.
5. As características de qualidade são elencadas acima das perguntas, e são representadas por cores, sendo as características de cor verde - Fatores de Desenvolvimento (Probabilidade) e as de cor amarela são referentes aos Fatores de Negócio (Impacto).

Conforme apresentado nas seções 2.2.2, os modelos de qualidade permitem avaliar softwares de acordo com diferentes aspectos e comumente representam as características desejáveis a um software em uma estrutura hierárquica. Dessa forma, as perguntas apresentadas nessa funcionalidade, de acordo com a Figura 4.26, refletem

de forma objetiva as subcaracterísticas da qualidade que são definidas pela ISO/IEC 25010 [2] e que servirão como atributos de avaliação de probabilidade e impacto da má qualidade do software.

| FATORES | CARACTERÍSTICAS DE QUALIDADE | PERGUNTAS |
|---|---|---|
| FATORES DE DESENVOLVIMENTO (PROBABILIDADE) | Adequação Funcional | Qual a probabilidade, da funcionalidade não atender aos objetivos específicos do usuário a que foi destinado? |
| | Adequação Funcional | Qual a probabilidade da funcionalidade apresentar erros? |
| | Manutenalibildade | Qual a probabilidade de não existir documentação nas funcionalidades? |
| | Manutenabilidade | Qual a probabilidade da funcionalidade permitir que uma modificação seja implementada causando defeitos no produto existente? |
| | Manutenabilidade | Qual a probabilidade, da funcionalidade permitir alterar elementos do software com impacto mínimo? |
| | Segurança | Qual a probabilidade, da funcionalidade permitir que uma modificação seja implementada causando defeitos no produto existente? |
| | Segurança | Qual a probabilidade, da funcionalidade permitir vazamento de informações sigilosas? |
| | Segurança | Qual a probabilidade, da funcionalidade permitir que um usuário acesse dados não autorizados? |
| | Confiabilidade | Qual a probabilidade da funcionalidade não responder, quando requerida ao uso em condições normais? |
| | Confiabilidade | Qual a probabilidade, da funcionalidade não restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha? |
| | Usabilidade | Qual a probabilidade, da funcionalidade não ser adequada a necessidade dos usuários? |
| | Usabilidade | Qual a probabilidade da funcionalidade, não permitir ao usuário operá-lo de forma fácil? |
| | Usabilidade | Qual a probabilidade, da funcionalidade permitir que o usuário cometa falhas? |
| | Usabilidade | Qual a probabilidade da funcionalidade não está adaptável à usuários com diferentes características e habilidades? |
| | Compatibilidade | Qual a probabilidade da funcionalidade não estar compatível com outros recursos de Softwares ou Hardwares? |
| | Portabilidade | Qual a probabilidade, da funcionalidade não ser de fácil utilização em outro ambiente? |
| | Eficiência de desempenho | Qual a probabilidade, da funcionalidade não fornecer tempo de resposta e processamento apropriados quando o software executa suas funções, em alta sazonalidade? |
| FATORES DE NEGÓCIO (IMPACTO) | Eficácia | Qual o Impacto, caso a funcionalidade não possibilite ao usuário atingir seu objetivo com precisão? |
| | Eficiência | Qual o impacto, caso a funcionalidade não possibilite ao usuário atingir metas especificadas com completeza? |
| | Satisfação | Qual o impacto, caso a funcionalidade não seja satisfatória ao usuário? |
| | Ausência de Risco | Se a funcionalidade falhar, temos impacto de danos à vida? |
| | Ausência de Risco | Se a funcionalidade falhar, temos impacto de danos à imagem ao negócio? |
| | Ausência de Risco | Se a funcionalidade falhar, temos impacto de danos financeiro? |
| | Cobertura de contexto | Qual o impacto caso a funcionalidade falhe em seu objetivo? |

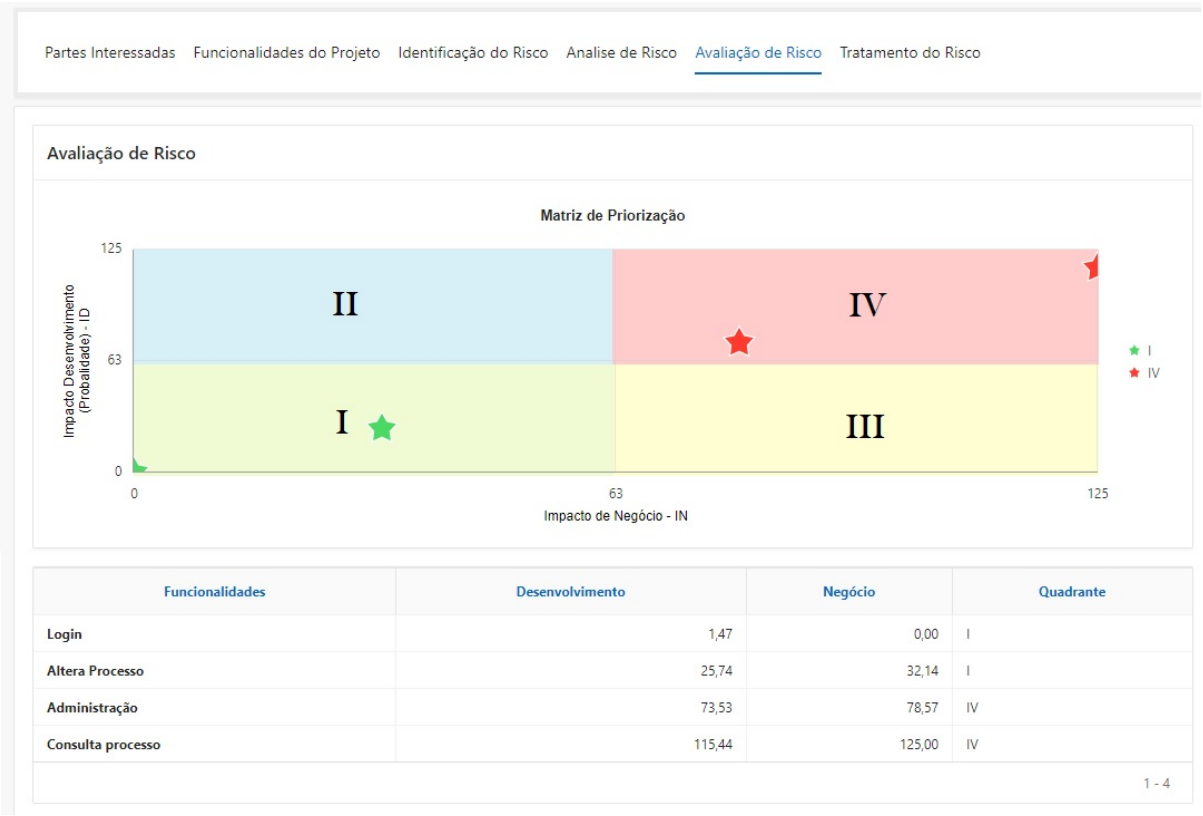
Figura 4.26: Perguntas das Características de Qualidade para Avaliação do Produto de Software

Fonte: Adaptado pela autora [2, 17]

Para cada funcionalidade apresentada deve ser realizado uma análise qualitativa, por meio da pontuação de cada pergunta elencada nesta tela. Cada pergunta deve receber uma pontuação de 0 a 5, conforme apresentado na Figura 4.10.

Uma vez realizada a análise qualitativa, a ferramenta realiza o cálculo do Score do Risco que é representada na funcionalidade de "Avaliação do Risco", conforme detalhado a seguir. Neste sentido, considerando o processo de gestão de risco para Qualidade de software, essa funcionalidade abrange a Fase de Análise de Riscos disposto 4.3.4 deste trabalho.

- **Avaliação de Risco:** Permite a visualização da matriz de risco seguido do *Score* de Desenvolvimento e Negócio e o quadrante em que a funcionalidade foi enquadrada, conforme apresentada na Figura 4.27.




| Funcionalidades | Desenvolvimento | Negócio | Quadrante |
|---|---|---|---|
| Login | 1,47 | 0,00 | I |
| Altera Processo | 25,74 | 32,14 | I |
| Administração | 73,53 | 78,57 | IV |
| Consulta processo | 115,44 | 125,00 | IV |

1 - 4

Figura 4.27: Matriz de Priorização - Avaliação do Risco
Fonte: Elaborado pela autora

A matriz é representada por meio de 4 quadrantes e é entreposta no quadrante, a depender da análise de risco que foi realizada para cada funcionalidade, conforme listado a seguir:

- I - Baixo Impacto/Baixa Probabilidade.
- II - Baixo Impacto/Alta Probabilidade.
- III - Alto Impacto/Baixa Probabilidade
- IV- Alto Impacto/Alta Probabilidade

Os Scores, são calculados, a partir dos seguintes conceitos e fórmulas:

**SD – Score do Desenvolvimento**, são calculados a partir das pontuações das perguntas da probabilidade, que neste contexto abrange as perguntas referente a:

Adequação funcional, Confiabilidade, Usabilidade, Eficiência de desempenho, Manutenibilidade, Portabilidade, Compatibilidade, Segurança. A fórmula é representada conforme a Equação 4.3:

$$\sum P * Pe/QPP \tag{4.3}$$

**SN - Score de Negócio,** são calculados a partir das pontuações das perguntas de Impacto, que neste contexto abrange as perguntas referente a: Eficácia, Eficiência, , Ausência de Risco e Cobertura de contexto. A fórmula é representada conforme a Equação 4.4:

$$\sum I * Pe/QPI \tag{4.4}$$

As equações são representadas pelas seguintes legendas:

- P - Probabilidade
- I - Impacto
- Pe - Peso
- QPP - Quantidade de Perguntas Probabilidade diferentes de 0
- QPI - Quantidade de Perguntas de Impacto diferentes de 0

A matriz apresenta uma escala de 125 no eixo y por 125 no eixo x, sendo o primeiro representado pela probabilidade de ocorrência no desenvolvimento e a segunda representada pelo impacto no negócio. As funcionalidades são elencadas nos quadrantes com o resultado de SD e SN, por meio de estrelas que apresentam os nomes das funcionalidades e seus devidos resultados, conforme apresentado na Figura 4.28.


Figura 4.28: Funcionalidades nos quadrantes da matriz de risco
Fonte: Elaborado pela autora

Essa matriz suportará a decisão da priorização dos testes a serem realizados, seguindo as seguintes perguntas:

1. Qual teste atacar primeiro e o que atuar depois?
2. Quais funcionalidades devo priorizar?

Considerando o processo de gestão de risco para Qualidade de software, essa funcionalidade abrange o processo e atividades da Fase de Análise de Riscos disposto 4.3.5 deste trabalho.

- **Tratamento de Risco:** Permite a visualização da estratégia de teste elencada para o produto, com vistas a mitigar falhas nas funcionalidades e assegurar o atendimento das características de qualidade do software. A estratégia é baseada na análise qualitativa realizada nas funcionalidades que representam o processo de avaliação de risco (Identificação e Risco, Análise de Risco e Avaliação de Risco).

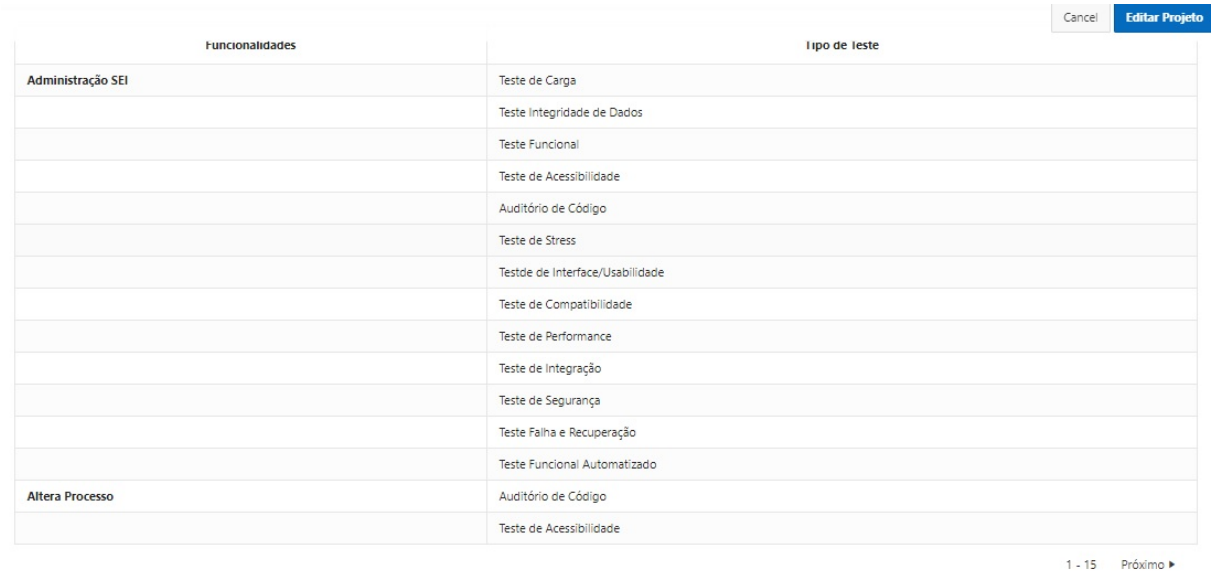



Figura 4.29: Tela de Tratamento do Risco
Fonte: Elaborado pela autora

Nesta tela, para cada funcionalidade analisada, são sugeridos tipos de testes a serem realizados. Esses tipos de testes são compostos pela pontuação realizada na Análise de Risco para cada pergunta. Para cada característica de qualidade apresentada 2.2, os tipos de testes podem ser realizados a depender no nível de criticidade da probabilidade de ocorrência e do impacto.

O processo de tratamento de risco para qualidade de Software considera os tipos de testes. Foram construídas duas matrizes que atribuem para cada pergunta relativa

às características de qualidade, um tipo de teste a depender da pontuação realizada na "Análise de Risco", conforme apresentadas nas Figuras 4.30 e 4.31.

| Características de Qualidade - Fatores de Desenvolvimento (PROBABILIDADE) | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tipos de testes a serem realizados para reduzir probabilidade de ocorrência de falhas nas funcionalidades | Adequação funcional | | Manutenabilidade | | | Segurança | | | Confiabilidade | | Usabilidade | | | | Compatibilidade | Portabilidade | Eficiência de desempenho |
| | Pergunta 1 | Pergunta 2 | Pergunta 3 | Pergunta 4 | Pergunta 5 | Pergunta 6 | Pergunta 7 | Pergunta 8 | Pergunta 9 | Pergunta 10 | Pergunta 11 | Pergunta 12 | Pergunta 13 | Pergunta 14 | Pergunta 15 | Pergunta 16 | Pergunta 17 |
| Teste Funcional | 3 à 4 | 3 à 4 | | | | | | | | | | | 3 à 5 | | | | |
| Teste Funcional Automatizado | 5 | 5 | | 4 e 5 | | | | | | | | | | | | | |
| Teste Exploratório | 1 à 2 | 1 à 2 | 1 à 5 | | | | | | | | | | 1 à 2 | | | | |
| Teste de Segurança | | | | | | 4 à 5 | 4 à 5 | 4 à 5 | | | | | | | | | |
| Auditório de Código | | | | | 3 à 5 | | | | | | | | | | | | |
| Teste de Acessibilidade | | | | | | | | | | | | | | 3 à 5 | | | |
| Teste de Interface/Usabilidade | | | | | | | | | | | 3 à 5 | 3 à 5 | | | | | |
| Teste de Carga | | | | | | | | | | | | | | | | | 4 à 5 |
| Teste de Stress | | | | | | | | | | | | | | | | | 5 |
| Teste de Performance | | | | | | | | | 3 à 5 | | | | | | | | |
| Teste falha e recuperação | | | | | | | | | | 3 à 5 | | | | | | | |
| Teste integridade de Dados | | | | | | 3 à 5 | | | | | | | | | | | |
| Teste de compatibilidade | | | | | | | | | | | | | | | 4 à 5 | 4 à 5 | |
| Teste de Integração | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

Figura 4.30: Matriz - Fatores de Desenvolvimento (Probabilidade)
Fonte: Elaborado pela autora

Considerando o contexto apresentado a Figura**??**, pode receber pontuação de escala de 1 a 5, que são provenientes da aba da pontuação elencadas para a Análise de Riscos. Ou seja, o tipo de teste a ser provido pela ferramenta, irá depender da pontuação elencada.

São previstos os principais tipos de testes que devem se realizados, para ser buscar qualidade de Software, conforme características previstas na ISO25010.

Para o tipo de teste preenchido com um 'X', é necessário a realização de testes em toda o sistema, uma vez que sempre é necessário realizar teste nas integrações do produto de software.

| Características de Qualidade - FATORES DE NEGÓCIO (IMPACTO) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Tipos de testes a serem realizados para reduzir o impacto de falhas nas funcionalidades | Eficácia | Eficiência | Satisfação | Ausência de Risco | | | Cobertura de contexto |
| | Pergunta 1 | Pergunta 2 | Pergunta 3 | Pergunta 4 | Pergunta 4 | Pergunta 6 | Pergunta 7 |
| Teste Funcional | | 3 à 5 | | x | 3 à 4 | 3 à 4 | 3 à 4 |
| Teste Funcional Automatizado | | | | x | 5 | 5 | 5 |
| Teste Exploratório | 1 à 2 | | | x | 1 à 2 | 1 à 2 | 1 à 2 |
| Teste de Segurança | | | | x | | | |
| Auditório de Código | | | | x | | | |
| Teste de Acessibilidade | | | | x | | | |
| Teste de Interface/Usabilidade | | | 3 à 5 | x | | | |
| Teste de Carga | | | | x | | | |
| Teste de Stress | | | | x | | | |
| Teste de Performance | | | | x | | | |
| Teste Falha e Recuperação | | | | x | | | |
| Teste Integridade de Dados | | | | x | | | |
| Teste de Integração | | | | x | | | |
| Teste de compatibilidade | | | 3 à 5 | x | | | |

Figura 4.31: Matriz - Fatores de Negócio (Impacto)
Fonte: Elaborado pela autora

A Figura 4.31, apresenta todos os tipos de teste voltados a qualidade em uso, entretanto existe uma condição de que, caso seja elencanda pontuação para a pergunta listada a seguir, todos os tipos de testes deverão ser realizados:

– Se a funcionalidade falhar, temos impacto de danos à vida?

A matriz foi gerada com base nos conceitos apresentados no referencial teórico, considerando os tipos de teste demonstrados 2.3 deste trabalho.

O tratamento de risco apresenta uma sugestão de atuação para reduzir o risco da má qualidade do Software por meio do tipo de teste, que é mais bem detalhado em seu processo, 4.3.7 deste trabalho.

- Monitoramento e Análise Crítica: Permite a visualização dos indicadores da execução do projeto de teste, em que é apresentada uma visão geral dos tipos de teste que estão sendo realizados e todos os defeitos identificados. As Figuras 4.32 e 4.33, apresentam as duas visões que são apresentadas na ferramenta. Os indicadores apresentados nesta funcionalidade foram realizados por meio da ferramenta *Power BI*.

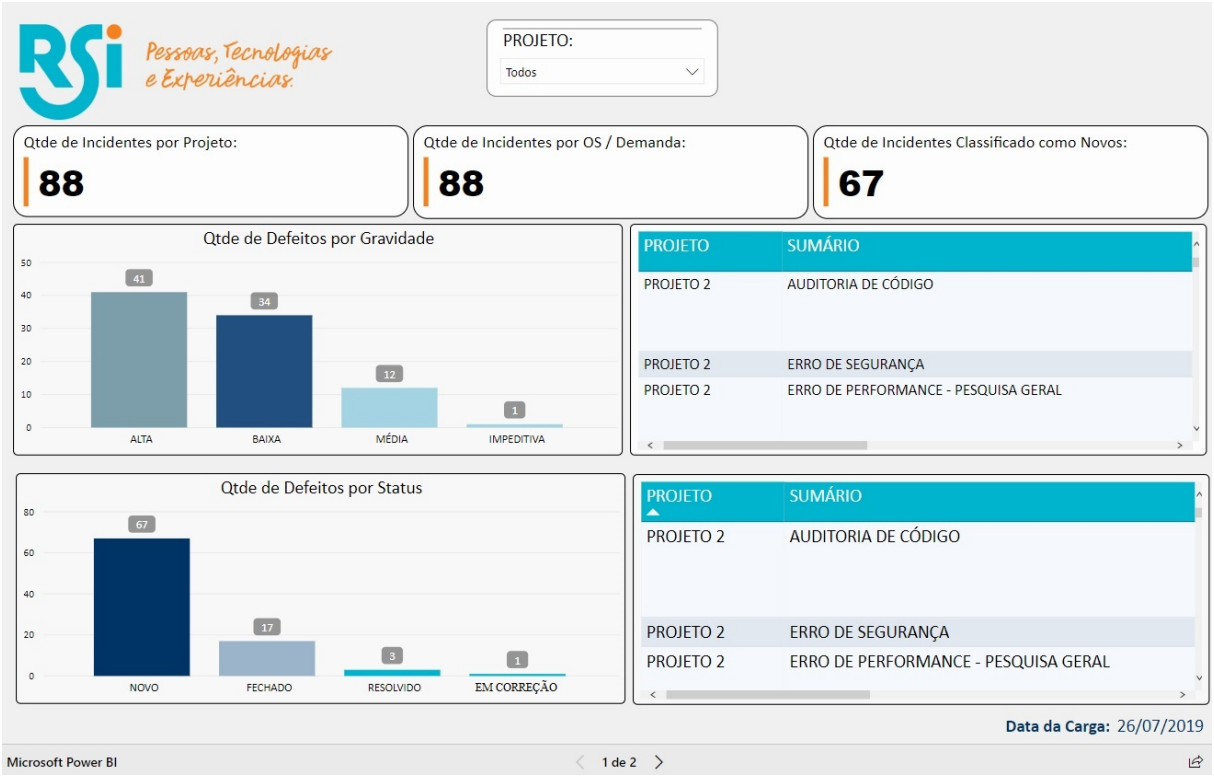



Figura 4.32: Tela 01 de Indicadores de Acompanhamento dos Testes de software
Fonte: Elaborado pela autora

O painel 01 apresenta uma visão de indicadores, conforme detalhamento a seguir. As legendas para melhor entendimento dos status, são destacadas no Anexo III desta pesquisa:

- Quantidade de defeitos por gravidade, apresenta todos os defeitos identificados por meio dos testes de software e gravidade de cada um dos defeitos identificados. Podem ser apresentadas as gravidades Impeditiva, Alta, Média e Baixa. Essa visão permite uma análise na gravidade dos erros que estão sendo abertos;
- Quantidade de defeitos por status, apresenta todos os defeitos identificados por meio dos teste de software e qual o status do defeito naquele momento, podendo ser atribuído os seguintes status: Pendente, Em Correção, Resolvido, Retrabalho, Validado e Fechado. Essa visão permite um acompanhamento de como está o status do erros naquele momento;
- Ao lado do gráfico é apresentado o resumo dos defeitos identificados;
- Ao clicar no gráfico, é realizado um filtro da seleção.

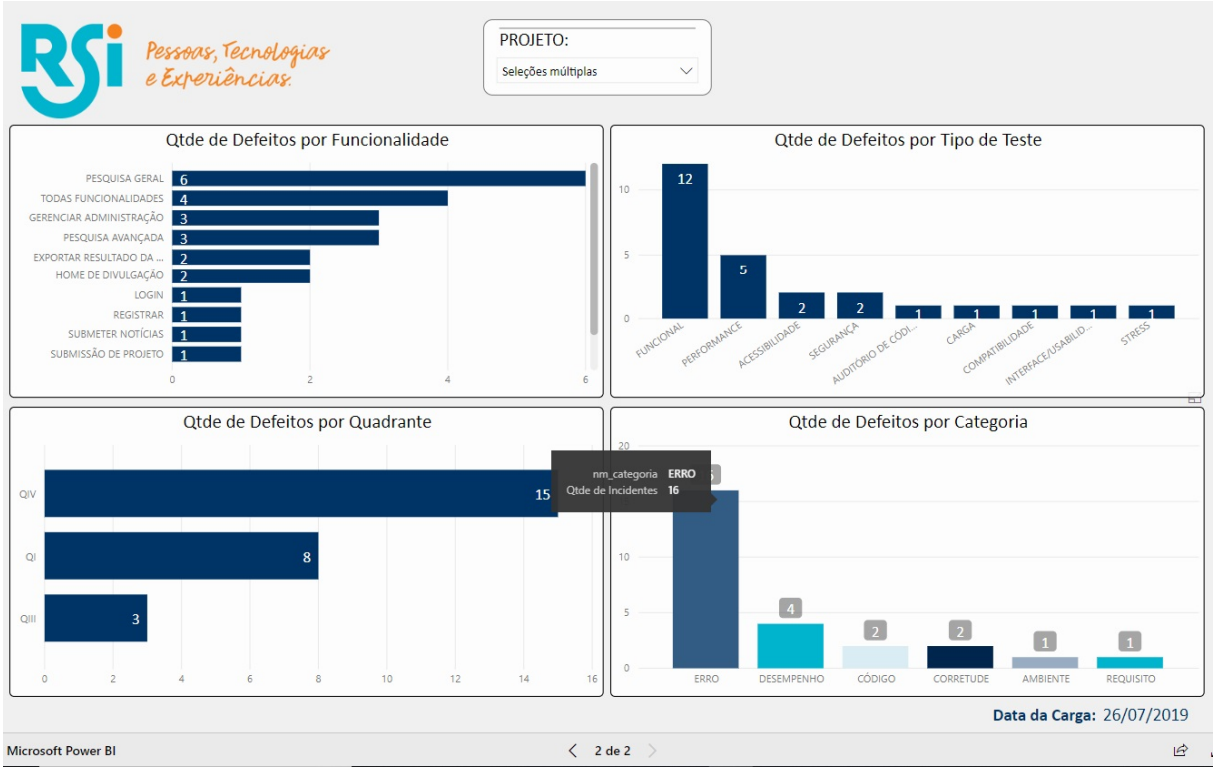



Figura 4.33: Tela 02 de Indicadores de Acompanhamento dos Testes de software

A painel 02 apresenta uma visão de indicadores, conforme detalhamento abaixo:

- Quantidade de defeitos por funcionalidades, ao qual serão expostas as funcionalidades e quantitativos de erros para cada uma delas.
- Quantidade de defeitos por tipo de teste, apresenta a quantidade de defeitos reportados por tipo de teste. Os tipos de testes apresentados, são os mesmos indicados nas matrizes apresentadas 4.2.3, na funcionalidade de tratamento de risco.
- Quantidade de defeitos por quadrante, sendo apresentadas as quantidade de defeitos por quadrante de acordo com o Resultado da Análise de Risco;
- Quantidade de defeitos por categoria, apresenta a categoria do erro identificado, podendo abranger: Erros de ambiente, requisitos, código, banco de dados, massa de teste, sistema, desempenho, segurança.
- Ao clicar no gráfico, é realizado um filtro da seleção.

O Anexo IV, apresenta qual o objetivo da medição de cada um dos indicadores apresentados nos 2 painéis, bem como, qual a fonte de coleta, qual a base de cálculo, qual a meta a ser atingida, para que se tenha uma entrega de qualidade e ainda a forma de apresentação de cada indicador.

Os indicadores permitirão um Monitoramento e uma Análise Crítica baseados nos problemas que serão identificados durante o teste de software, de acordo com a estratégia definida pela ferramenta proveniente das funcionalidades de Avaliação dos Riscos e Tratamento dos Riscos. São executadas cargas diárias de atualização dos indicadores, através de um sincronismo com a ferramenta de gestão de incidentes Mantis.

Uma vez descrita todas as funcionalidades que compõem a ferramenta, segue descrição dos atores da ferramenta e seus principais papéis e responsabilidades, junto ao processo de aplicação.

- Administrador: Papel responsável pela administração da ferramenta que concede acesso aos demais atores e ainda permite a criação da estrutura de cadastro da ferramenta.
- Gerente de negócio e time: Papel responsável pela aplicação da Gestão de Risco para Qualidade de software da ferramenta.
- Gerente/líder de Teste: Papel responsável pela alimentação das informações do projeto de teste e aplicação do método junto e permite a criação da estrutura de cadastro da ferramenta.

Uma vez detalhados todos os requisitos, será apresentado na próxima seção toda a arquitetura da ferramenta de gestão de risco para Qualidade de software.

### 4.2.3 Análise e Design

Esta fase compreendeu o desenho da ferramenta, definição da arquitetura, a escolha das tecnologias e definição do modelo entidade relacionamento, que permitiu a apresentação do modelo de dados para descrever os aspectos de informação de um domínio de negócio ou seus requisitos de processo, conforme apresentado 4.2.2.

Arquitetura

O *Oracle Application Express - APEX* tem sua arquitetura definida pela sua instalação junto ao banco de dados *Oracle*, composto por dados em tabelas e código *PL/SQL- Procedural Language/Structured Query Language*[101].

Seu acesso é dado pelo navegador, que envia uma solicitação de URL, que é convertida na chamada apropriada do Oracle Application Express (APEX) em Procedural Language/Structured Query Language (PL/SQL). Depois que o banco de dados processa a procedure, os resultados são retornados ao seu navegador como Hypertext Markup Language (HTML) . Esse ciclo acontece toda vez que é solicitada ou enviada uma nova página.[101]

O estado da sessão do aplicativo é gerenciado nas tabelas do banco de dados no *Oracle Application Express*. Não usa uma conexão de banco de dados dedicada. Em vez disso, cada solicitação é feita por meio de uma sessão de banco de dados separada, consumindo recursos mínimos da CPU- Unidade Central de Processamento. [101]

A aplicação reside completamente em um Banco de Dados *Oracle* que é composto por aproximadamente 215 tabelas e 200 objetos PL/SQL contendo mais de 300.000 linhas de código.

Sua arquitetura é composta pela Figura 4.34, que compreende os itens abaixo:

- Roda no *Oracle HTTP Server* (que contém o *Apache*) com o módulo *(Oracle PL/SQL Toolkit)*;
- É dividida em 3 camadas: Web *Browser* + *Apache* com + Banco de Dados *Oracle com Apex Engine*.

Como requisito para instalação do *Browser* (Cliente):

- Deve suportar *JavaScript*, *HTML* 4.0 e *CSS 1.0*; Cookies devem estar habilitados;
- Recomenda-se o uso do Internet Explorer 6.0 (ou superior) ou Firefox 1.0 (ou superior);

Os 3 principais componentes do Apex são: *Application Builder*, *SQL Workshop* e Utilitários *(Data Workshop)*, conforme descritos abaixo:

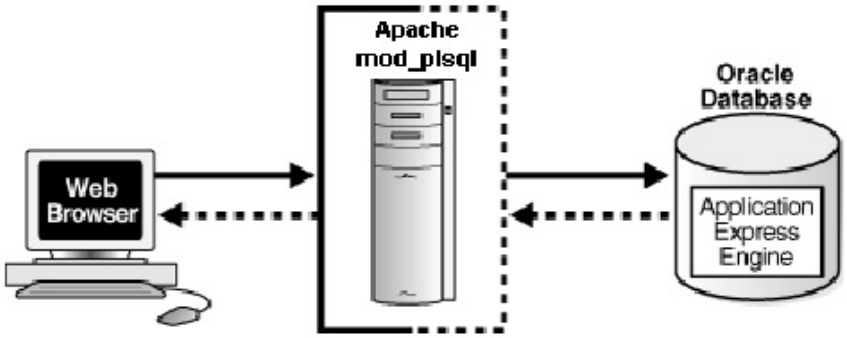



Figura 4.34: Arquitetura APEX Oracle

- *Application Builder*: principal componente de desenvolvimento, possibilita criar aplicações web, gerar scripts das aplicações, definir e configurar linguagem das aplicações.
- *SQL Workshop*: permite a criação e gerenciamento dos objetos de Banco de Dados-BD, executar e carregar instruções e scripts *SQL-Structured Query Language*.
- Utilitários: permite ler e extrair dados do Banco de Dados Oracle, monitorar o BD, gerar scripts DDL - Linguagem de definição de dados e ainda ver relatórios e views do Apex etc.

A Ferramenta de Gestão de Risco para Qualidade de software, teve seu desenvolvimento pautado na arquitetura apresentada na versão *Education*, que está disponibilizada para professores e alunos. Neste sentido, a próxima seção, apresentará o modelo de dados criado para a ferramenta apresentada nesta pesquisa.

### Modelo Entidade Relacionamento

O Modelo Entidade Relacionamento - MER, é o modelo conceitual utilizado pela Engenharia de software para descrever os objetos que são transcritos pelas entidades envolvidas em um domínio de negócios, com suas características por meio dos atributos e seus devidos relacionamentos. Neste sentido, essa seção apresenta a representação abstrata de estrutura de banco de dados da ferramenta de Gestão de Risco para Qualidade de software, conforme Figura 4.35.

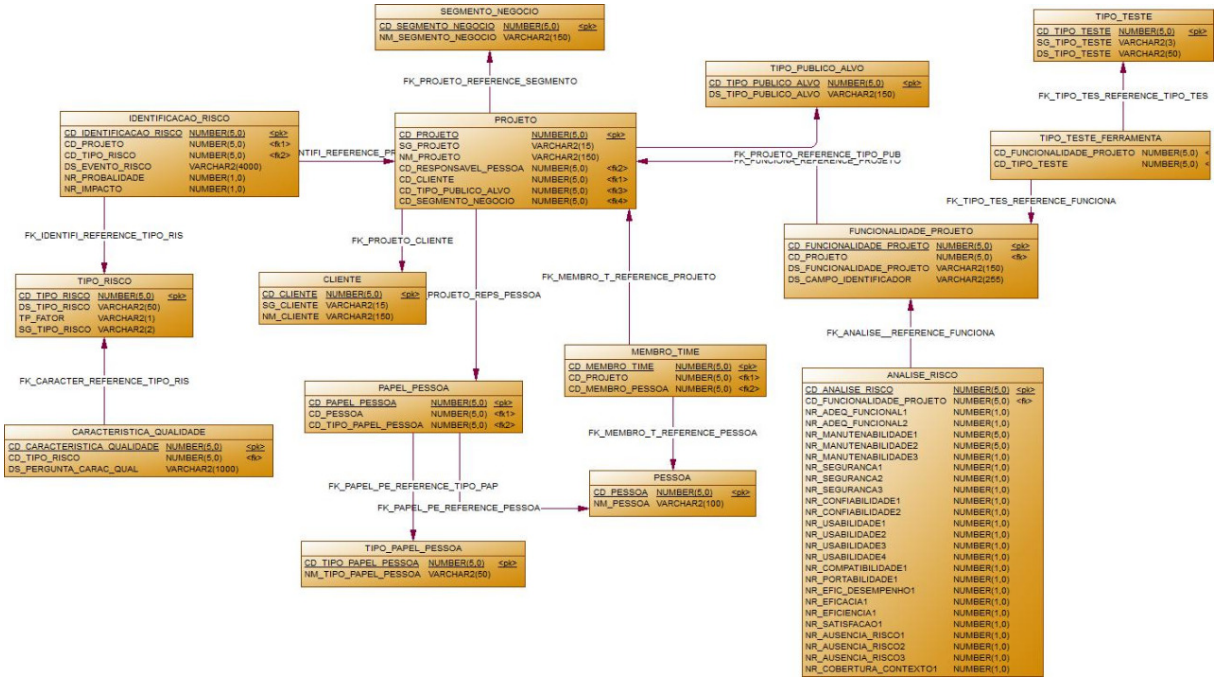



Figura 4.35: Modelo Entidade Relacionamento - Ferramenta de Gestão de Risco para Qualidade de software

Fonte: Elaborado pela autora

O modelo de dados foi concebido através da ferramenta Power Designer, conforme descrito 4.2.1 e implantado junto ao Banco de dados Oracle por meio da ferramenta APEX.

### 4.2.4 Construção

Esta fase foi conduzida, através da construção da ferramenta, por meio do software APEX - Oracle Application Express, conforme apresentado na Figura 4.36.

Trata-se de uma ferramenta de desenvolvimento rápido de pequenas de médias aplicações Web Oracle Database-Centric. O desenvolvimento foi realizado por meio de um browser, na utilização de templates e construção de PL/SQL.

Considerando a forma de desenvolvimento, nenhum código é gerado. Foram criados metadados e armazenados no banco de dados ORACLE e são utilizados por uma série de PL/SQL packages que geram páginas Hypertext Markup Language (HTML). Ao se criar uma funcionalidade, Apex, por meio de um código PL/SQL renderiza as páginas das aplicações em tempo real. O Browser envia um REQUEST que é traduzido em uma chamada APEX PL/SQL. O Banco de Dados processa o PL/SQL e os resultados são retornados para o browser como Hypertext Markup Language (HTML) . A aplicação reside completamente em um BD Oracle que é composto por aproximadamente 215 tabelas e 200 objetos PL/SQL contendo mais de 300.000 linhas de código.

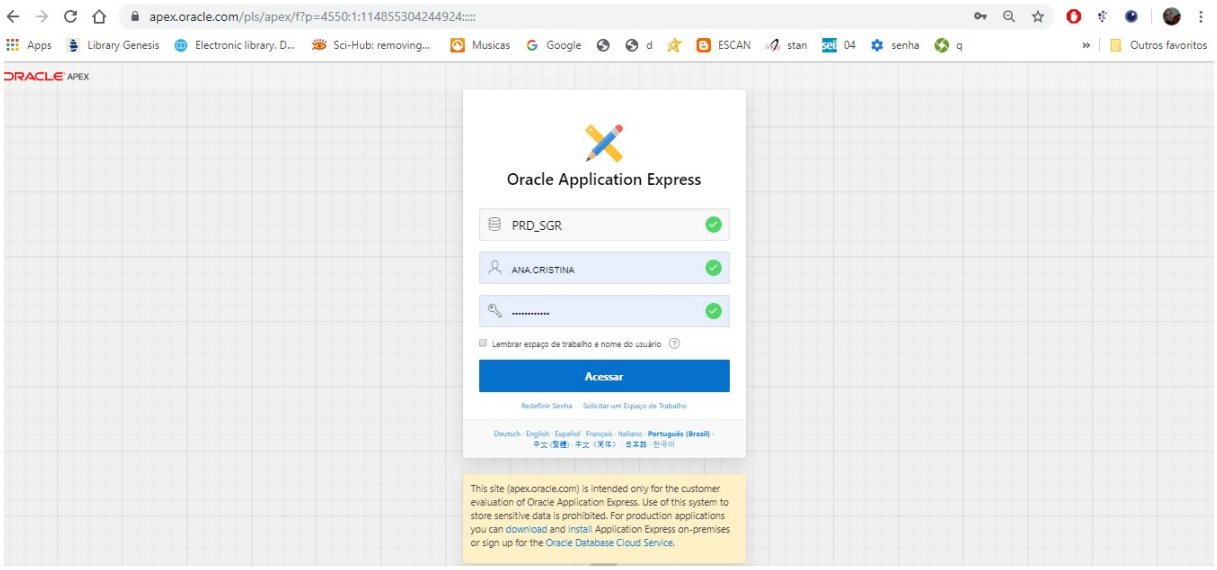



Figura 4.36: Ambiente de Desenvolvimento da Ferramenta de Gestão de Risco para Qualidade de Software

Fonte: Elaborado pela autora

Considerando o exposto, a Figura 4.37 apresenta a interface de desenvolvimento da ferramenta que foi utilizada para a construção da ferramenta.

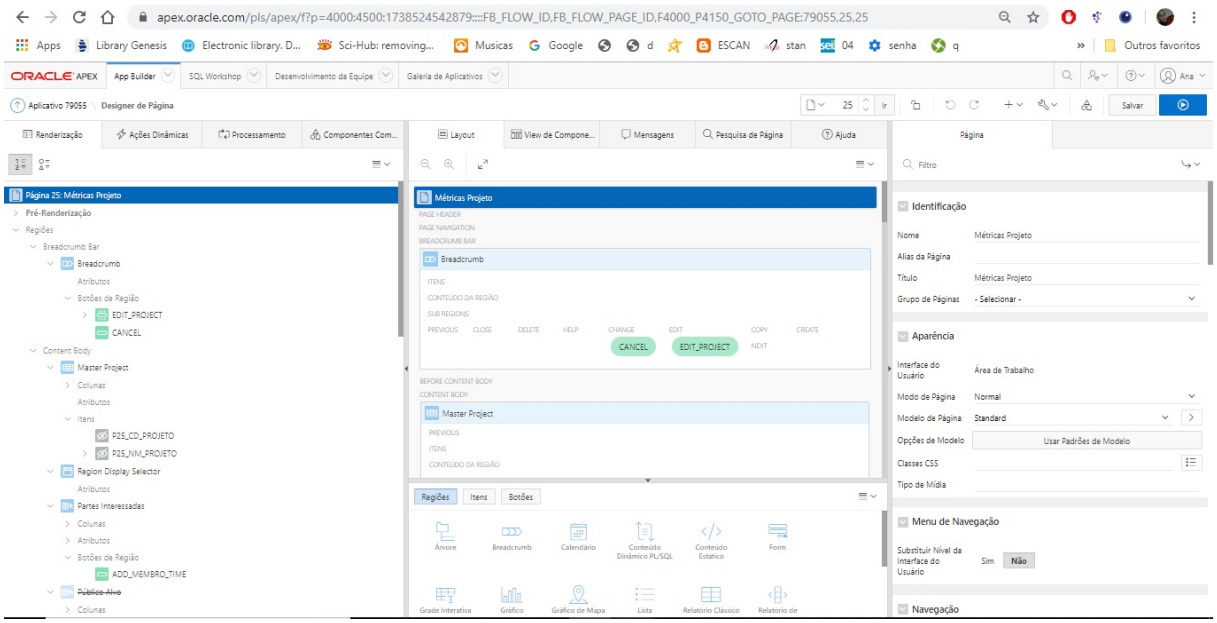



Figura 4.37: Interface do Ambiente de Desenvolvimento da Ferramenta de Gestão de Risco para Qualidade de Software

Fonte: Elaborado pela autora

Como vantagens, foram identificados, os pontos abaixo listados na utilização desta ferramenta de desenvolvimento.

- Facilidade de aprendizado e de desenvolvimento;
- Funcionalidades para facilmente gerar backups das aplicações (incluindo páginas web e objetos de BD);
- Possui relatórios para análise de desempenho e auditoria de acesso de workspaces, aplicações, páginas etc;
- Possibilidade de criar aplicações web rapidamente, utilizando diversos templates e wizards;

Uma vez concluído o desenvolvimento, foram realizados ciclos de teste para validação da implementação, conforme descrito na próxima seção 4.2.5.

### 4.2.5 Testes

A fase de teste foi aplicada ao longo do desenvolvimento da ferramenta proposta, a fim de realizar os testes funcionais ou teste da "caixa-preta", em que foram implementadas as regras constantes 4.2.2 e foram realizadas simulações para entender o comportamento da aplicação durante a navegação do usuário, ou seja, testando definitivamente a funcionalidade do sistema, reproduzindo um cenário de produção e identificando possíveis problemas.

A identificação e correção dos erros identificados durante as fases de teste, foram controladas por meio da ferramenta Trello, conforme apresentado na Figura 4.38.

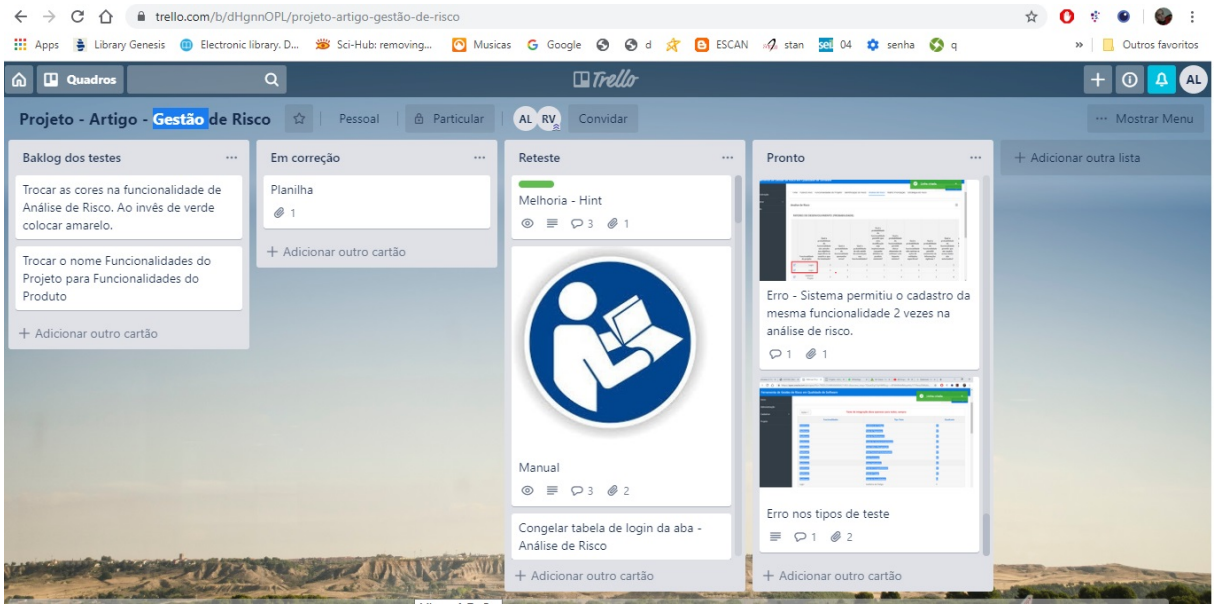



Figura 4.38: Controle dos testes
Fonte: Elaborado pela Autora

A raia de “Backlog dos testes”, representou todos os defeitos identificados durante a fase de testes. A raia “Em correção”, expôs todos os defeitos em correção pela autora. A raia “Reteste”, representou todos os defeitos, já corrigidos para a realização de um novo teste, para verificação se o defeito foi devidamente corrigido. A raia “Pronto” apresentou todos os defeitos corrigidos e retestados, prontos para serem implementados na ferramenta. Uma vez concluído a fase de teste de toda a ferramenta, ela foi disponibilizada em ambiente de produção, conforme apresentado na próxima seção. Durante esta fase, foram identificados um total de 12 defeitos e 13 melhorias.

### 4.2.6 Implantação

Esta fase compreendeu a elaboração do manual da ferramenta, constante no Anexo I desta pesquisa, e a atividade de liberação, em que foi disponibilizada em ambiente de produção para devida utilização. A ferramenta foi disponibilizada por meio da URL apresentada na Figura 4.39.

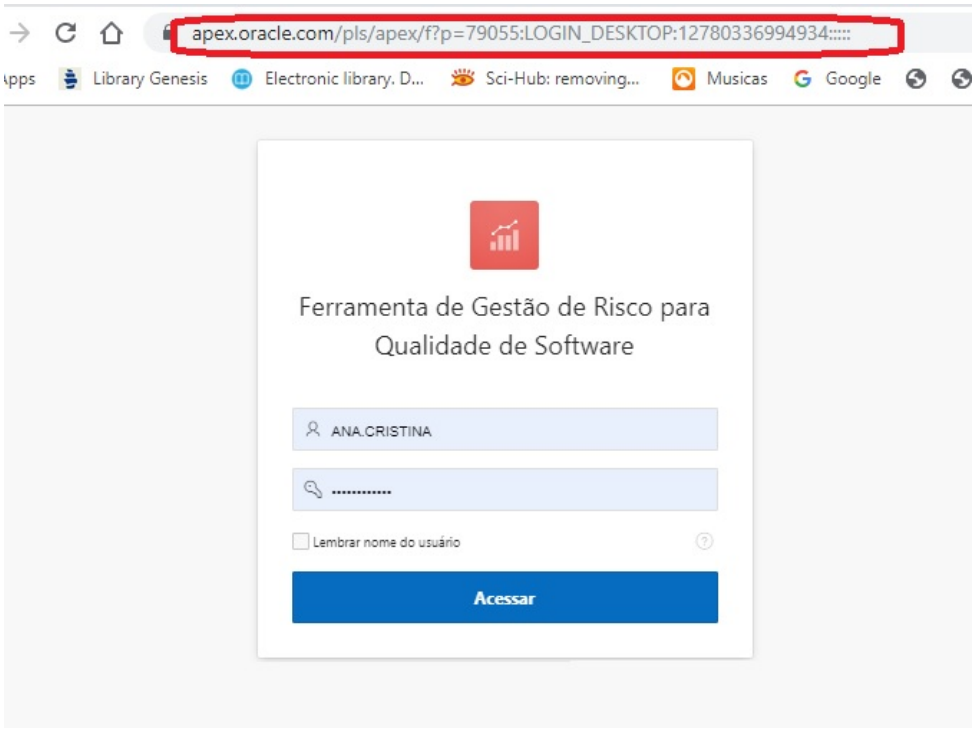



Figura 4.39: URL da ferramenta de Gestão de Riscos para Qualidade de Software
Fonte: Elaborado pela Autora

Desta forma, uma vez apresentadas todas as fases de construção da ferramenta proposta, serão apresentadas na próxima seção as fases de aplicação da ferramenta e detalhamento de suas atividades.

## 4.3 Fases de Aplicação da Ferramenta

Esta seção apresentará as fases de aplicação da ferramenta proposta, que foram estruturadas utilizando como base o processo de gestão de risco da ISO31000[1]. A Figura 4.40 expõem a relação entre as fases de gestão de risco da ISO e as fases de aplicação da ferramenta proposta nesta pesquisa por meio do processo de Gestão de Risco para Qualidade de software, que serão detalhadas a partir da seção 4.3.1.

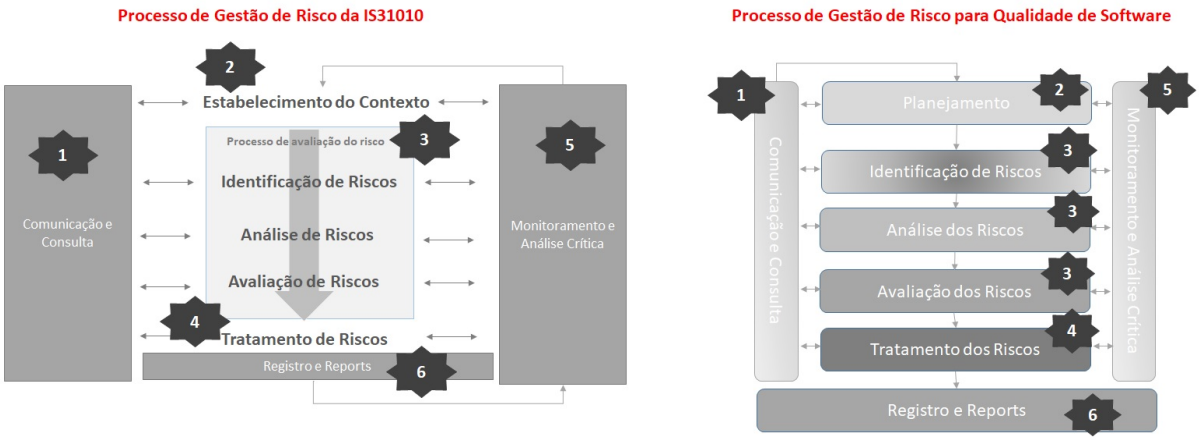



Figura 4.40: Fases do Processo de Gestão de Risco
Fonte: ISO3100 adaptada pela autora

Neste sentido o processo de gestão de risco para Qualidade de software, detalha as seguintes fases:

### 4.3.1 Fase de Comunicação e Consulta

Essa fase inclui atividades para Comunicar e Consultar as partes interessadas sobre o processo de gestão de risco em qualidade e software. Busca auxiliar as partes interessadas na compreensão do objetivo da aplicação da ferramenta e método estabelecido. A Comunicação, busca promover a conscientização e o entendimento da aplicação do método através da ferramenta, enquanto a Consulta busca envolver o retorno e informação para auxiliar a aplicação da ferramenta e tomada de decisão. Esta fase, é composta pelas seguintes atividades, que incluem mais não se limitam a:

- Workshop ou reunião para apresentação da Ferramenta de Gestão de Risco para Qualidade de software, explicação dos conceitos e papéis e responsabilidades no processo;
- Identificação inicial das partes interessadas;

A comunicação e consulta, é uma fase recorrente que ocorre em todas as etapas do processo de gestão de risco da Qualidade de Software e trata-se uma atividade externa à ferramenta, que dará subsídio a todo o processo. Uma vez realizada a fase de comunicação e consulta, em que todas as partes interessadas tem conhecimento da aplicação da ferramenta, são necessários a realização do planejamento e estabelecimento do contexto de aplicação da ferramenta, conforme apresentado na próxima seção.

### 4.3.2 Fase de Planejamento

Essa fase inclui atividades de Planejamento, Estabelecimento de Contexto e Estabelecimento de Escopo para aplicação da ferramenta de Gestão de Risco para Qualidade de software, com objetivo de estabelecer o escopo por meio dos contextos externo e interno. A aplicação da ferramenta está voltada ao contexto da Qualidade do Produto de software, assim sendo, é necessário identificar as informações gerais do produto a ser avaliado. Esta fase, é composta pelas seguintes atividades, que incluem, mais não se limitam a:

- Avaliar ativos de processos organizacionais, tal como planos, processos, políticas, procedimentos e bases de conhecimento específicos da organização para a Qualidade de Software;
- Identificação das informações necessárias que possibilitem uma compreensão adequada do contexto de aplicação e do produto a ser avaliado, com vistas a subsidiar a execução das próximas etapas, podendo compreender a documentação de requisitos do produto e documento de arquitetura do Software;
- Identificação das partes interessadas que irão participar do processo de gestão de risco, sendo necessário escolher as pessoas apropriadas e com conhecimento no Software em avaliação, necessária também a participação de técnicos para atuação na probabilidade e negócio para atuação no impacto;
- Identificação das informações do Produto de Software a ser avaliado, sendo elas: Segmento de Negócio, Público Alvo, Sigla do Projeto, Nome do Projeto, Responsável pelo Projeto e Organização;
- Identificação dos Itens de riscos que são representados pelas funcionalidades/requisitos do software em avaliação. Este deve ser de comum compreensão a todas as partes interessadas do processo.

A fase de Planejamento, ocorre no início do processo, entretanto é incremental, podendo ser atualizado a qualquer momento a depender da necessidade do projeto. É representada na ferramenta por meio das Abas de Cadastro, Criação do Projeto, Partes Interessadas e Funcionalidades do Projeto, conforme descrito 4.2.3 do módulo de Gestão de Risco. Uma vez identificado o contexto de aplicação, escopo e identificação das partes interessadas, é iniciada a fase de gestão de risco da ferramenta de Gestão de Risco para Qualidade de Software, conforme apresentado na próxima seção.

As próximas 3 fases englobam o processo de avaliação de risco, conforme apresentado na Figura 4.40, sendo elas a identificação de riscos, análise de riscos e avaliação de riscos. Este processo deve ser conduzido pelo Gerente ou Líder de Teste de forma sistemática e iterativa junto aos stakeholders do produto de Software.

### 4.3.3 Identificação de Riscos

Esta fase inclui atividades de identificação de risco que tem como objetivo reconhecer e descrever riscos que possam impedir a qualidade do produto de Software. Pode ser utilizada, uma variedade de ferramentas e técnicas para identificar incertezas que podem afetar um ou mais objetivos do produto de software. Deve ser realizado junto aos stakeholders específicos definidos neste processo, conforme apresentado na Figura 4.41

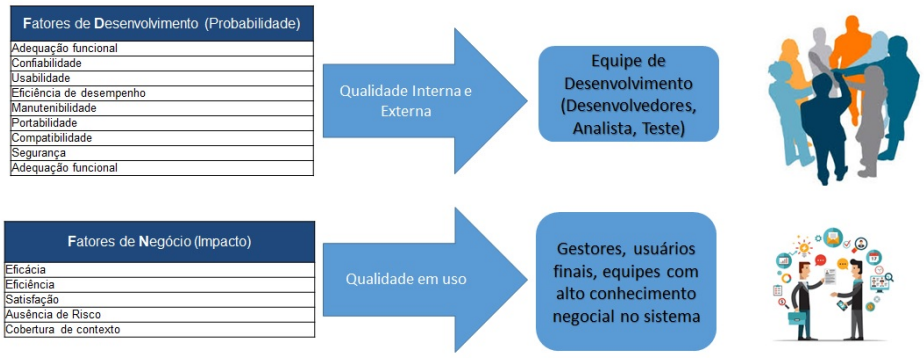



Figura 4.41: Processo de identificação do risco com os *stakeholders*
Fonte: Elaborado pela autora

Os riscos identificados nesta fase, impactam diretamente a próxima fase, uma vez que os riscos elencados refletem pesos na fase de Análise de Risco. É composta pelas seguintes atividades, que incluem, mais não se limitam a:

- Identificação do(s) evento(s) de riscos que podem afetar a Qualidade do Produto de Software.
- Classificação técnica dos eventos de riscos, que neste contexto, englobam todas as características de qualidade de um software, previsto na ISO25010[2], conforme detalhado 2.2.2.
- Pontuação da Probabilidade de ocorrência do risco, levando em consideração a escala prevista na Figura 4.10. Essa pontuação deve ser elencada pelo time de desenvolvimento que pode envolver: Equipe de Requisitos, Equipe de Desenvolvimento, Gerente do projeto, Equipe de Teste, Arquitetura entre outros que participam do projeto à frente do desenvolvimento do produto, conforme apresentado na Figura 4.41.
- Pontuação do Impacto de ocorrência do risco, levando em consideração a escala prevista na Figura 4.10. Essa pontuação deve ser elencada pelos gestores de negócio que podem envolver a área demandante do produto de software ou técnicos que conhecem o produto, conforme apresentado na Figura 4.41.

- Uma vez elencando o evento(s) de risco(s), devem ser pontuado obrigatoriamente a probabilidade de ocorrência e impacto, caso ocorra.

Para a identificação do evento de risco, podem ser utilizadas as seguintes ferramentas e técnicas, conforme apresentada nas seções 2.4 e 2.6.1, que incluem, mais não se limitam a:

- Reuniões
- *Brainstorming*
- Entrevistas estruturadas ou semiestruturada
- *Delphi*
- Listas de verificação
- *SWIFT - Structured What If Technique*
- Análise de cenário
- Análise de impacto nos negócios:
- *Inside-Out*
- *Outside –in*

É representada na ferramenta pela aba de "Identificação de Risco"conforme descrito na seção 4.2.3 no módulo de gestão de risco. A fase de Identificação de risco é incremental, sendo necessária sua atualização sempre que surgirem novos riscos ou os já existentes deixarem de existir. Uma vez identificados os riscos a próxima fase é a Análise de Risco, conforme descrito a seguir.

### 4.3.4 Análise de Riscos

Esta fase, inclui atividades de Análise de risco que tem como propósito compreender a natureza dos riscos e suas características. Assim sendo, esta fase engloba uma análise qualitativa, por meio da pontuação da probabilidade de ocorrência e impacto nos itens de risco, que são representados pelas funcionalidades/requisitos do Software a ser avaliado. Esta análise é realizada por meio de perguntas embasadas nas características de qualidade interna e externa e qualidade em uso da ISO25010[2]. A análise de riscos pode ser influenciada por qualquer divergência de opiniões, vieses, percepções do risco e julgamento, uma vez que são realizadas junto ao time de técnico do projeto e gestores de negócio. Esta fase é composta pelas seguintes atividades, que incluem, mais não se limitam a:

- Pontuação das perguntas de Probabilidade de ocorrência da falha no software e impacto, junto aos itens de riscos (Funcionalidades/Requisitos), levando em consideração a escala prevista na Figura 4.10.
- Listagem de todas as funcionalidades do produto de software que são os fatores de riscos. Este formulário deve ser aplicado a todos as funções/requisitos que compõem um projeto de software.
- Probabilidade e impacto são variáveis independentes, que serão identificadas e avaliadas, levando em consideração as características de qualidade, através das perguntas que são mapeadas na ferramenta, sendo que:

  **Probabilidade - Riscos Técnicos** é a possibilidade ou chance de um evento de risco ocorrer em um produto de software que devem ser identificados pela equipe de desenvolvimento, conforme apresentado na Figura 4.41.

  **Impacto - Riscos de Negócio** é o efeito no produto de software se o evento ocorrer e deve ser identificado pelos usuários de negócio, conforme apresentado na Figura 4.41.
- Reunião de consenso, caso existam divergências entre as pontuações atribuídas às perguntas, sendo que todas as partes interessadas, são consideradas igualmente importantes.

É representada na ferramenta pela da aba "Análise de Risco"descrita na seção 4.2.3 módulo de gestão de risco. Esta fase fornece entrada para a fase de avaliação de riscos, para decisão, se o risco necessita ser tratado e qual estratégia e métodos mais apropriados para tratamento do risco. Trata-se de uma fase incremental, sendo necessária sua atualização sempre que surgirem novos riscos ou os já existentes deixarem de existir. Uma vez realizada a Análise de risco, a próxima fase é a Avaliação de Riscos, conforme apresentada na próxima seção.

### 4.3.5 Avaliação de Riscos

Esta fase inclui atividades de avaliação de risco, com objetivo de apoiar à decisão. A avaliação envolve a análise de uma matriz de riscos, que é representada por meio de quadrantes, nos quais os itens de riscos são enquadrados, com base na avaliação realizada na fase anterior de Análise de Risco. Esta fase é composta pelas seguintes atividades, que incluem, mais não se limitam a:

- Tratamento do risco a depender do quadrante em que o item de risco (funcionalidade/requisito) for enquadrada;

- Análises adicionais para melhor entendimento;
- Priorização para a realização dos testes dos itens de risco, com base nos quadrantes apresentados.

Considerando a qualidade do software, ser o foco da ferramenta, os itens de riscos (Funcionalidades/Requisitos) encaixados nos quadrantes poderão ter as seguintes consequências, caso ocorram erros: I – Insignificantes, II – Menores ,III – Importantes e IV - Graves.

A ferramenta apresenta a matriz considerando a probabilidade de ocorrência para riscos técnicos e Impacto considerando riscos de negócio conforme apresentado na Figura 4.42 abaixo:

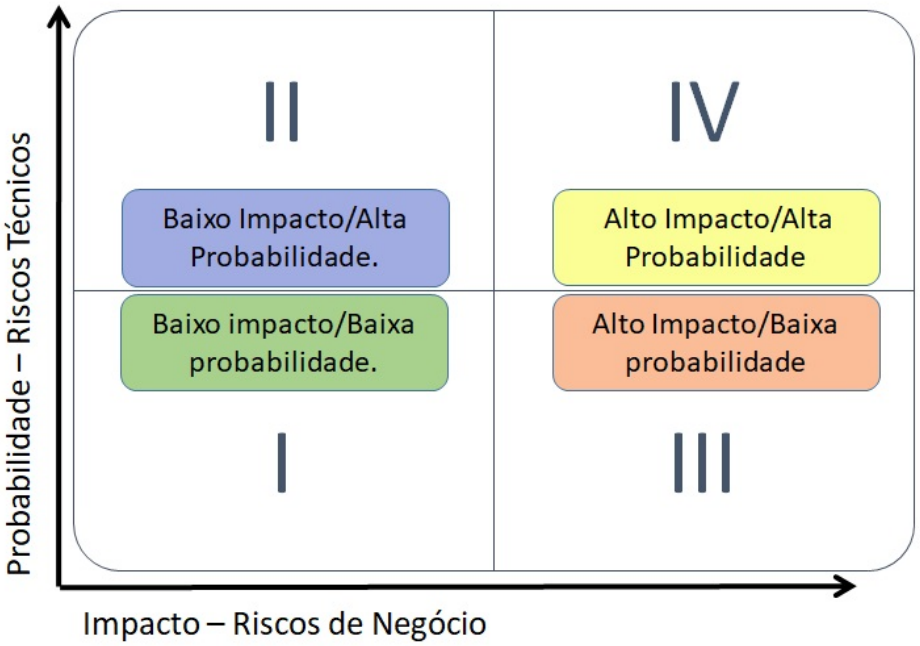



Figura 4.42: Matriz de Risco da Avaliação do Risco
Fonte: Elaborado pela autora

Essa fase é representada na ferramenta, pela aba "Avaliação de Riscos", descrita na seção 4.2.3 do módulo de gestão do risco. Convém que o resultado da avaliação de riscos seja registrado, comunicado e então validado pelos stakeholders participantes do processo de aplicação da ferramenta de gestão de risco para Qualidade de software. Esta fase ocorre, sempre que tiver atualização na fase anterior de Análise de Risco. Uma vez realizada a Avaliação de Riscos a próxima fase é o tratamento de riscos, descritos a seguir.

### 4.3.6 Tratamento de Riscos

Esta fase inclui atividades de tratamento de risco, que têm como objetivo selecionar e implementar opções para abordar os riscos. Neste contexto a ferramenta apresenta uma sugestão de quais tipos de teste realizar, para cada funcionalidade/requisito do produto avaliado, com objetivo de reduzir os riscos da má qualidade do Software. Esta fase é composta pelas seguintes atividades, que incluem, mais não se limitam a:

- Formular e selecionar os tipos de testes a serem aplicados no produto de Software;
- Balancear os benefícios potenciais para o alcance dos objetivos, face aos custos, esforço ou desvantagens da implementação;
- Planejar e implementar o tratamento do risco;

É representada na ferramenta, pela aba "Tratamento de Riscos"descrito na seção 4.2.3. Convém que o resultado do tratamento do Risco, seja avaliado quanto ao custo benefício da aplicação dos testes sugeridos pela ferramenta de gestão de risco para a qualidade de software. Esta fase é incremental e ocorre sempre que houver atualização junto a fase de Avaliação de Riscos. Uma vez concluída a Avaliação do Risco, é necessário o monitoramento e análise crítica, conforme descrito na próxima seção.

### 4.3.7 Monitoramento e Análise Crítica

Esta fase inclui atividades de monitoramento e análise crítica, com vistas a assegurar e melhorar a qualidade e eficácia da implementação dos resultados do processo de gestão de risco da qualidade de Software. A ferramenta prevê indicadores de gestão de teste para o monitoramento da qualidade do Software e para a gestão dos defeitos identificados durante a fase de teste. Esta fase é composta pelas seguintes atividades, que incluem, mais não se limitam a:

- Monitoramento e análise crítica, periódica do processo de gestão de riscos e seus resultados;

- Monitoramento e análise crítica em todos os estágios do processo;
- Avaliação da eficácia deste tratamento;
- Incluem planejamento, coleta e análise de informações, registro de resultados e fornecimento de retorno;

É representada pela ferramenta, através da aba "Monitoramento e Análise Crítica"descrito na seção 4.2.3 módulo de indicadores. Apresenta os seguintes resultados:

- Quantidade de defeitos por Gravidade;
- Quantidade de defeitos por Status;
- Quantidade de defeitos por Tipo de Teste;
- Quantidade de defeitos por Categoria;
- Quantidade de defeitos por funcionalidade;
- Quantidade de defeitos por prioridade do quadrante.

Convém que os resultados do monitoramento e análise crítica sejam incorporados em todas as atividades de gestão de desempenho, medição e relatos voltados a gestão de risco para a Qualidade de Software. Dado esta fase, é necessário manter o registro e *reports*, conforme explicado na próxima fase.

### 4.3.8 Registro e Reports

Essa fase inclui atividades de Registro e Reports que tem como objetivo documentar o processo de gestão de risco para Qualidade de Software e seus resultados por meio de mecanismos apropriados. Esta fase é composta pelas seguintes atividades, que incluem, mais não se limitam a:

- Comunicar atividades e resultados às partes interessadas do processo de gestão de risco para Qualidade de software;
- Fornecer informações para tomada de decisão;
- Melhorar as atividades de gestão de riscos;

É representada na ferramenta, por todas as abas contidas nela, que são descritas na seção 4.2.3 que apoiam a registro de todo o processo de gestão de risco para Qualidade de Software, podendo ser acessada a qualquer momento, tendo histórico armazenado para ser utilizado como base de conhecimento.

Finalizada a apresentação da ferramenta de Gestão de Risco para a Qualidade do Software, o próxima seção, apresenta como ocorre a iteração das fases de aplicação da ferramenta junto às fases de desenvolvimento de software.

## 4.4 Aplicação da ferramenta em ambientes de desenvolvimento de Software

Conforme apresentado na seção 2.1 do referencial teórico, são vários os modelos de processo de desenvolvimento de Software e saber qual utilizar vai depender da necessidade de cada projeto. Assim sendo, a ferramenta apresentada nessa pesquisa, foi construída para ser utilizada por qualquer modelo da engenharia de Software durante às fases de desenvolvimento de Software aplicada atreladas as fases de teste de Software.

O processo idealizado para aplicação da ferramenta nas fases de desenvolvimento de software por ser visualizado na Figura 4.43:

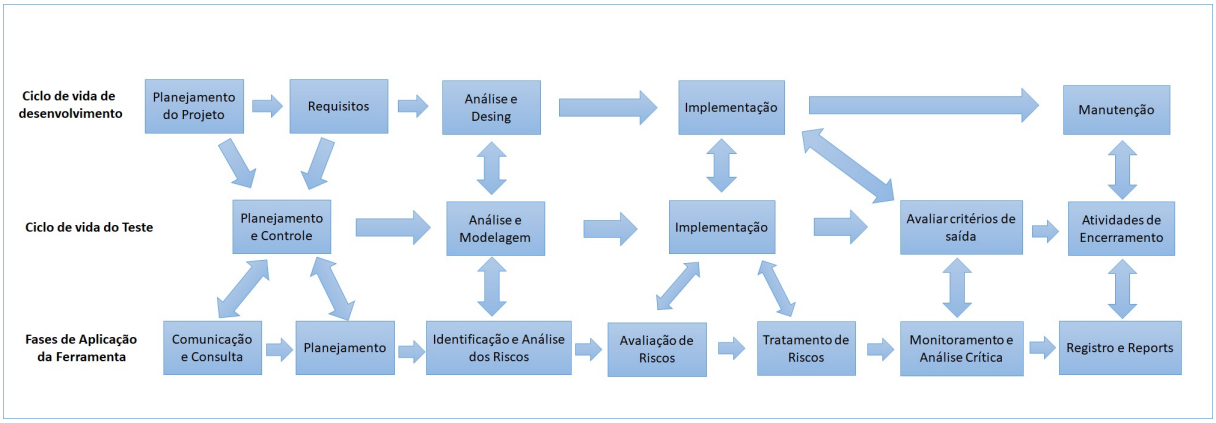



Figura 4.43: Relação entre o processo de desenvolvimento, teste e gestão de risco para Qualidade de Software

Fonte: Elaborado pela autora

A primeira camada, apresenta o ciclo de vida do desenvolvimento, que pode abranger qualquer metodologia, podendo ser iterativo incremental, cascata ou qualquer outro, ao qual é necessário destacar a fase de teste de software que é representado pelo item 2. A segunda camada, por sua vez, apresenta o processo de teste de software, exibido na seção 2.3.1, com suas respectivas fases, ao qual a ferramenta deve ser provocada na fase de planejamento, uma vez que uma das principais saída é a estratégia de teste e matriz de priorização, que precisam ser previstas na etapa de planejamento. Já a camada 3, representa o processo de Gestão de Risco para Qualidade Software apresentado na seção 4.3 e fará interação direta com as demais fases do processo de teste de software que por consequência fará a iteração com as fases de desenvolvimento.

A interação ocorrerá ao momento que o processo de desenvolvimento de software for acionado, e por consequencia a fase de teste de software, que por sua vez

Finalizada a apresentação da ferramenta de Gestão de Risco para a Qualidade do Software, o próximo capítulo apresenta a aplicação desta ferramenta através de estudos de casos.

# Capítulo 5

# Aplicação da ferramenta de Gestão de Risco para Qualidade de Software: Estudos de caso

Com o propósito de comprovar a efetividade da ferramenta de Gestão de Risco para Qualidade de software, foram realizados estudos de casos em 2 projetos de desenvolvimento de software em organizações distintas.

O primeiro produto avaliado, já estava no ambiente de pré-produção, disponibilizado a um grupo restrito e reduzido de usuários, com intuito de testarem o sistema desenvolvido e fornecer *feedback* quanto a utilização. Este foi escolhido, visto que já era sabido alguns erros reportados pelo grupo que realizou a utilização do software, com objetivo de compreender se a ferramenta indicaria estratégias corretas para o objetivo ao qual foi construída.

O segundo projeto, foi realizado durante o desenvolvimento de software, que foi construído utilizando práticas ágeis em um cliente da empresa RSI Informática.

Este capítulo tem, como objetivo retratar os resultados dos estudos de caso realizados. Está estruturado de acordo com as fases de aplicação da ferramenta constante na seção 4.3 deste trabalho, aliados com a ferramenta apresentada na seção 4.2.3.

Inicialmente é descrita a fase de execução da Fase de Comunicação e Consulta, realizada para Comunicar e Consultar as partes interessadas sobre o processo de gestão de risco para Qualidade de software. Em seguida é descrita a execução da Fase de Planejamento realizada para o estabelecimento do escopo para aplicação da ferramenta. Depois é descrita a execução da Fase de Identificação de Riscos, com objetivo de reconhecer e descrever riscos que possam impedir a qualidade do produto de software. Logo após, é descrita a execução da fase de Análise de Riscos, com propósito de realizar análise qualitativa por meio da pontuação da probabilidade de ocorrência e impacto nos itens de

risco. Posteriormente é descrita a execução da fase de Avaliação de Riscos, que tem como propósito o apoio a tomada de decisão por meio da matriz de risco. Seguidamente é descrita a execução da fase de Tratamento de Risco, que tem como objetivo selecionar e implementar opções para abordar os riscos por meio dos tipos de teste. Logo após é descrita a execução Monitoramento e Análise Crítica que tem como objetivo a assegurar e melhorar a qualidade e eficácia da implementação dos resultados do processo de gestão de risco da qualidade de software. Por fim, é descrita a execução da Fase de Registro e *Reports* que tem como objetivo documentar o processo de gestão de risco para Qualidade de Software e seus resultados por meio de mecanismos apropriados.

## 5.1 Estudo de caso - PUMA - Plataforma Unificada de Metodologia Ativas

### 5.1.1 Caracterização do Projeto

Este projeto, consiste na construção de uma Plataforma Unificada de Metodologia Ativa que está sendo desenvolvida por meio do Programa Aprendizagem para o 3º Milênio (A3M) do CEAD/UnB patrocinado pelo valor de 25 mil em bolsas a estudantes de engenharias de software, computação e produção. O desenvolvimento da Plataforma apoiará a melhoria do processo de avaliação das disciplinas Projetos de Sistemas de Produção (PSP's) do curso de Engenharia de Produção da Universidade de Brasília (EPR/UnB) e possibilitará a mensuração da eficácia do Abordagem Baseada em Pprojeto (PBL) no ensino de engenharia.. Assim sendo, a Plataforma será uma ferramenta de integração de todos os resultados e insumos dos projetos que ocorrem do quarto ao décimo semestre, a partir da recepção dos problemas oriundos dos agentes externos, até a conclusão das disciplinas [102]. O projeto pode ser melhor conhecido por meio do vídeo de apresentação do projeto no *link*: https://www.a3m.cead.unb.br/projetos/plataforma-unificada-de-metodologia-ativa-puma e na URL da própria plataforma, que pode ser acessada, por meio da URL: http://pumaunb.herokuapp.com/, além dos artigos publicados[102].

Trata-se de um sistema web, desenvolvido nas tecnologias: Node.js, PostgreSQL e React.js. A construção da ferramenta está dividida em três módulos, conforme Figura 5.1, sendo o foco do estudo de caso o módulo de Divulgação.


Figura 5.1: Módulos - Plataforma Unificada de Metodologia Ativa-PUMA
Fonte: Elaborado pela autora

O módulo de divulgação abrangeu a construção das seguintes funcionalidades, conforme apresentado na Figura 5.2:

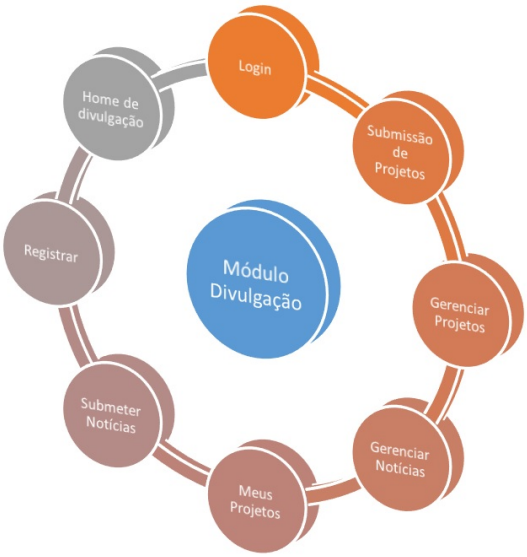



Figura 5.2: Funcionalidades do Módulos de Divulgação - PUMA
Fonte: Elaborado pela autora

As funcionalidades compostas neste módulo são abertas a todos os públicos de empresas privadas e públicas, pessoas físicas, alunos e professores, e tem como objetivo central a captação de projetos para serem trabalhados pelos alunos nas disciplinas de PSPs. Este módulo compõe desde do cadastro dos usuários externos, submissão e triagem dos projetos e divulgação dos projetos e notícias.

### 5.1.2 Fase de Comunicação e Consulta

Esta fase foi iniciada por meio da identificação das partes interessadas iniciais, que foi realizada por meio de uma reunião com a coordenadora do projeto Profa. Dra. Simone Borges. Neste sentido foi mapeado as partes interessadas, apresentada na Figura 5.3:

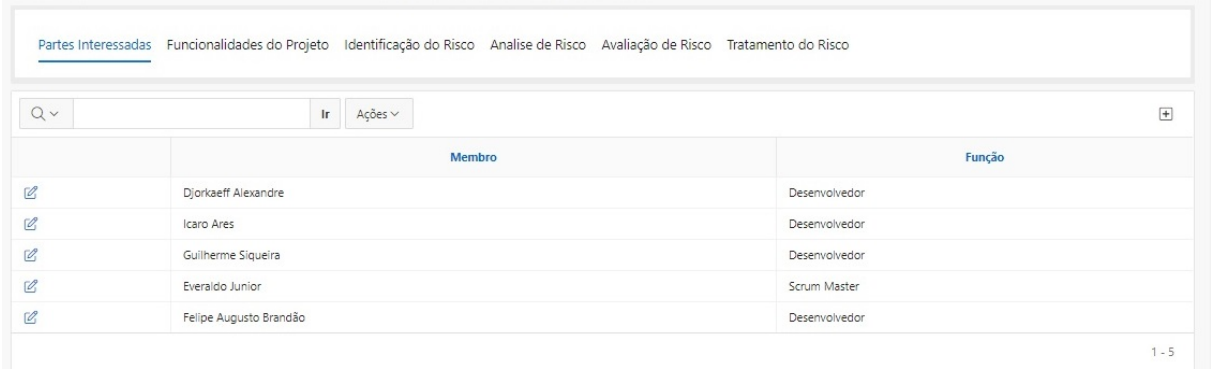



Figura 5.3: Fase de Comunicação e Consulta - Identificação das Partes Interessadas - PUMA

Fonte: Elaborado pela autora

Após a identificação das partes interessadas, foi realizado uma reunião para apresentação da ferramenta, com intuito de auxiliá-los na compreensão do objetivo da aplicação da ferramenta e método estabelecido. A reunião ocorreu em abril de 2019 e foi realizada no dia da própria aplicação da ferramenta.

### 5.1.3 Fase de Planejamento

Esta fase foi iniciada, por meio do estabelecimento do escopo para a aplicação da ferramenta de Gestão de Risco para Qualidade de software, através da identificação das informações gerais do produto em avaliação. Para esta fase, foi realizada as seguintes atividades:

- Avaliação da documentação do projeto;
- Avaliação da documentação de requisitos do PUMA;
- Definição das partes interessadas que irão participar do processo de Gestão de Risco para Qualidade de software;
- Identificação das informações do produto, sendo eles: Segmento de Negócio, Público Alvo, Sigla do Projeto, Nome do Projeto, Responsável pelo Projeto e Organização;
- Identificação dos Itens de riscos que são representados pelas funcionalidades/requisitos do software.

A Figura 5.4, apresenta todas informações levantadas para da fase de planejamento de acordo com as atividades listadas acima.

As informações acima detalhadas, refletem todas as funcionalidades constantes no módulo da plataforma e o identificador *UI - User Story*, representa Histórias de Usuário

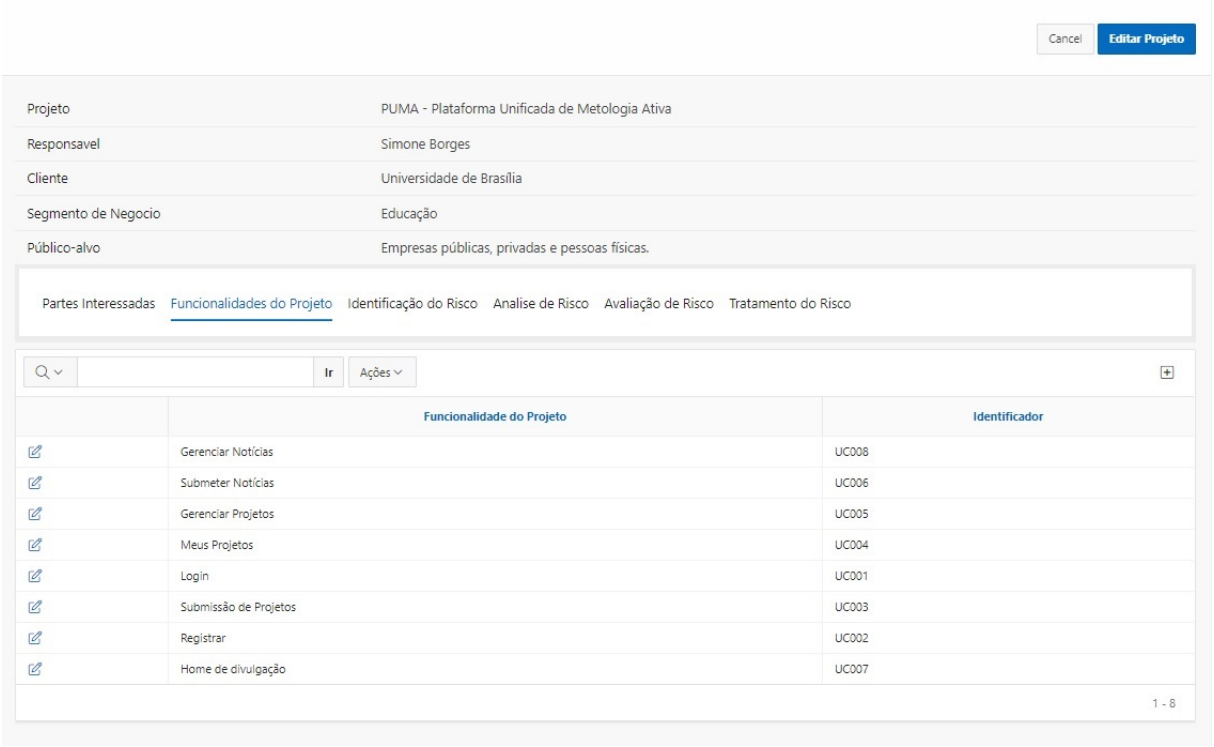



Figura 5.4: Fase de Planejamento - PUMA
Fonte: Elaborado pela autora

implantadas. Todos levantamentos foram realizados por meio da análise da documentação do projeto, que envolveu a proposta do projeto enviado ao programa A3M - Aprendizagem para o 3º milênio - UnB, o diagrama de caso de uso do projeto, com a representação visual das funcionalidades e a documentação de requisitos do projeto, constante na ferramenta Trello, conforme apresentada na Figura 5.5.

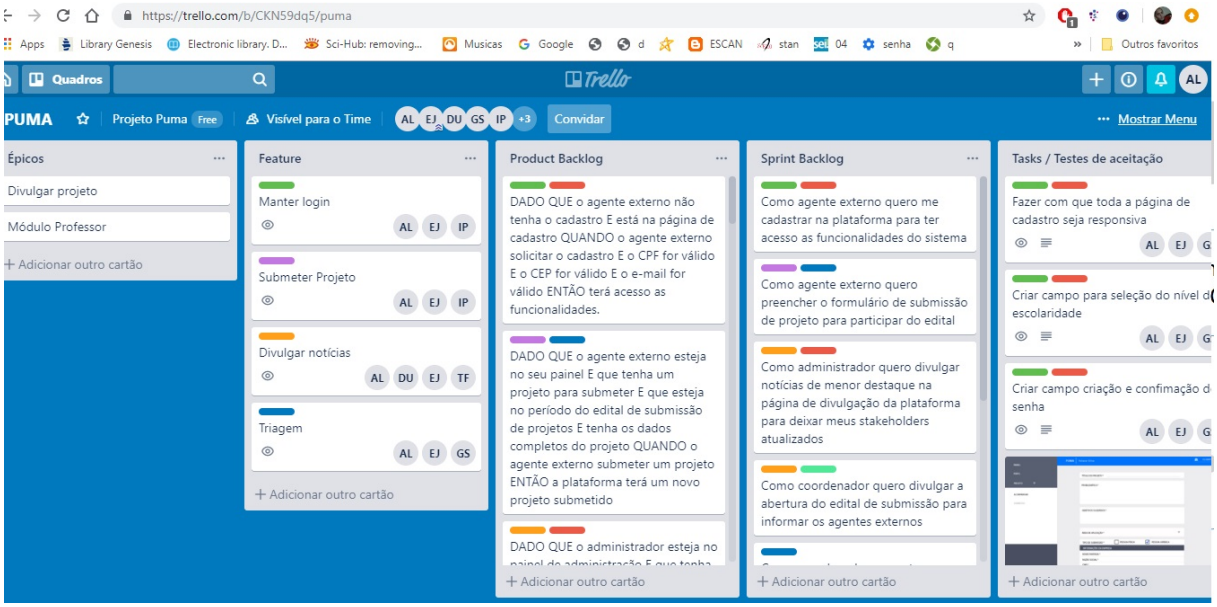



Figura 5.5: Trello do PUMA
Fonte: Elaborado pela equipe do projeto PUMA

Uma vez realizado todo o planejamento, as próximas fases englobam o processo de avaliação de risco, sendo elas a Identificação de Riscos, Análise de Riscos e Avaliação dos Riscos, conforme detalhamento a seguir.

### 5.1.4 Identificação de Risco

Esta fase foi iniciada, por meio de um *workshop* inicialmente junto a equipe técnica do projeto PUMA, com o objetivo de mapear os riscos que podem afetar a qualidade do produto de software, do ponto de vista do desenvolvimento e posteriormente junto a responsável do projeto com objetivo de identificar os riscos de negócio. Foi realizada seguindo os seguintes passos:

- Identificação dos eventos de riscos, que podem afetar a qualidade do produto de software;
- Categorização dos riscos identificados, por meio das características de qualidade;
- Classificação da probabilidade de ocorrência, por meio da pontuação dos riscos, seguindo a matriz de definição de probabilidade, conforme Figura 4.10;

A Figura 5.6, apresenta os riscos identificando junto a equipe, ao qual os riscos técnicos foram levantados pela equipe de desenvolvimento e os riscos de negócio foram mapeados pela responsável pelo projeto. Em seguida é realizado comentários sobre os riscos mapeados.

| | Característica de Qualidade que o evento de risco afeta | Evento Risco | Probalidade | Impacto |
|---|---|---|---|---|
| | Manutenibilidade | Ausência de requisitos funcionais | 3 | 3 |
| | Segurança | Falha de Segurança | 5 | 5 |
| | Eficiência de desempenho | Pico de utilização do software, além do esperado | 4 | 4 |
| | Satisfação | Não utilização do software pelos professores | 4 | 5 |
| | Usabilidade | Não entendimento dos usuários na utilização do software | 3 | 4 |
| | Manutenibilidade | Manutenção do software, posterior ao desenvolvimento | 4 | 5 |
| | Compatibilidade | Ausência de responsividade | 2 | 4 |
| | Usabilidade | Não utilização por deficientes visuais | 4 | 5 |
| | Portabilidade | Não utilização em qualquer plataforma | 4 | 4 |
| | Eficiência | Não atingimento ao objetivo com precisão | 2 | 5 |
| | Adequação funcional | Não submissão de projetos | 5 | 5 |
| | Ausência de Risco | Risco de imagem, caso não consiga enviar um projeto para UNB | 4 | 5 |
| | Eficiência de desempenho | Auto volume de armazenamento | 5 | 5 |
| | Manutenibilidade | Falta de conhecimento na linguagem ao qual o software foi desenvolvido | 3 | 3 |
| | Cobertura de contexto | Erro no software | 3 | 5 |
| | Eficácia | Não envio de projetos a UNB com edital aberto | 4 | 5 |
| | Confiabilidade | Não recuperação de dados em caso de erro | 4 | 4 |

1 - 17

Figura 5.6: Identificação dos Riscos - PUMA
Fonte: Elaborado pela autora

- A ausência de requisitos funcionais, foi identificado como risco, uma vez que não está sendo realizado a documentação do projeto de forma estruturada. A documentação está sendo realizada pela ferramenta Trello conforme apresentada na seção 5.1.3, por meio de história de usuário. Futuramente, pode-se ter problemas para manter o software.

- Falha de segurança, foi identificado como risco, uma vez que a equipe técnica do projeto sinalizou não está utilizando nenhuma prática voltada a desenvolvimento seguro. Futuramente pode-se ter problemas de segurança e de internalização junto ao Centro de Processamento de Dados (CPD) da Universidade de Brasília.

- Pico de utilização do software além do esperado, foi identificado como risco, uma vez que infraestrutura não está alinhada a necessidade do projeto. Foi reservado verba para compra de infraestrutura, entretanto não foi permitido a compra pela financiadora do projeto A3M, sendo necessário ação dentro do projeto para mitigar este risco.

- Não utilização do software pelos professores, foi mapeado como um risco pela gestora, uma vez que nem todos os professores estão apoiando o projeto e provendo informações. Considera-se de suma importância a utilização da plataforma por este público, visto que o objetivo central do projeto em medir eficácia da abordagem de ensino no curso de Engenharia de Produção da Universidade.

- Não entendimento dos usuários na utilização do software, foi mapeado como risco, uma vez que a interface não está padronizada.

- Manutenção do software posterior ao desenvolvimento, foi mapeado como um risco, uma vez que após a conclusão do projeto não tem equipe nem departamento responsável para assumir a sustentação do projeto. Foi elencado como um risco técnico voltado a manutenibilidade.
- Ausência de responsividade, foi apontado como um risco, uma vez que plataforma não foi desenvolvida, considerando a utilização em múltiplas plataformas, como por exemplo o mobile. Trata-se de um risco de um risco de compatibilidade.
- Não utilização por deficientes visuais, foi percebido como um risco, uma vez que a plataforma não está aderente os critérios de acessibilidade, considerando que a plataforma será aberta ao público geral. Foi enquadrado com um risco de usabilidade.
- Não utilização em algumas plataformas web, foi mapeado como um risco técnico de portabilidade, uma vez que a plataforma não está preparada para todos os *browsers*. Foi desenvolvida inicialmente apenas para o navegador *Chrome*.
- Não atingimento ao objetivo com precisão, foi mapeado como um risco de negócio, caso o usuário não consiga realizar todos os passos necessário na aplicação devido a erros, afetando assim a eficiência.
- Não submissão de projetos, foi identificado como um risco de negócio, uma vez que devido a erros, existe risco de não submissão. Este foi identificado, uma vez que no teste beta, alguns usuários não conseguiram realizar a tarefa completa, e por fim não submeteram os projetos.
- Risco de imagem, devido à má qualidade do software, foi apontado como um risco pelo negócio, uma vez que a má qualidade da plataforma pode ter danos à imagem da universidade.
- Auto volume de armazenamento, uma vez que infraestrutura não está alinhada a necessidade do projeto. Está sendo necessário algumas adaptações devido à falta de memória em disco para armazenamento dos arquivos do projeto. Foi enquadrado como um risco de Eficiência de desempenho.
- Falta de conhecimento na linguagem ao qual o software foi desenvolvido, foi mapeado, uma vez que alguns membros da equipe não têm domínio sobre a tecnologia empregada. Foi enquadrado como um risco de manutenibilidade.
- Erro no software, foi mapeado como um risco, uma vez que foram identificados vários erros na fase de teste. Foi classificado como risco que afeta a cobertura de contexto, uma vez que afeta o objetivo para ao qual o software foi desenvolvido na captação de agente externos;

- Não envio de projetos a UNB com edital aberto, foi detectado como risco de negócio, uma vez que o edital tem prazo de início e fim e não submissão de projetos pode afetar a quantidade de projetos a serem trabalhos pelos alunos dentro do semestre.
- Não recuperação de dados em caso de erro, foi identificado como um risco técnico, uma vez que o sistema não está tratando persistência de dados em caso de erros.

Uma vez realizada toda identificação dos riscos, categorizando quanto às características de qualidade que o risco afeta e ainda realizada a classificação da probabilidade de ocorrência e impacto, a ferramenta calculou os pesos conforme apresentado na Figura 5.7.


Figura 5.7: Pesos da Identificação dos Riscos - PUMA
Fonte: Elaborado pela autora

Os pesos calculados, serão atríbuídos as caracateristicas de qualidade, a serem utilizados na aba de Análise de Riscos.

### 5.1.5 Análise de Risco

Essa fase foi seguida após a fase de identificação de risco, por meio do mesmo workshop e reunião estabelecida na fase anterior com vistas a simplificar o processo de aplicação. Foi realizada junto a equipe técnica com a visão de desenvolvimento e responsável pelo projeto com a visão de negócio. Para esta fase, foram realizados os seguintes passos:

- Inclusão das funcionalidades que foram mapeadas por meio das "Funcionalidades do Projeto";
- Pontuação das perguntas relacionadas Fatores de Desenvolvimento (Probabilidade) junto ao time técnico, considerando a matriz de definição de probabilidade 4.10;
- Pontuação das perguntas relacionadas ao Fatores de Negócio (Impacto) junto a responsável pelo projeto, considerando a matriz de definição de probabilidade 4.10;
- Reunião de consenso para apresentação da pontuação realizada e ajustes;

A Figura 5.8 apresenta a tela da Análise de Risco, com o resultado do ensaio realizado, e a Figura 5.9 exibe o resultado completo da pontuação realizada pela equipe técnica e negócio da plataforma PUMA.

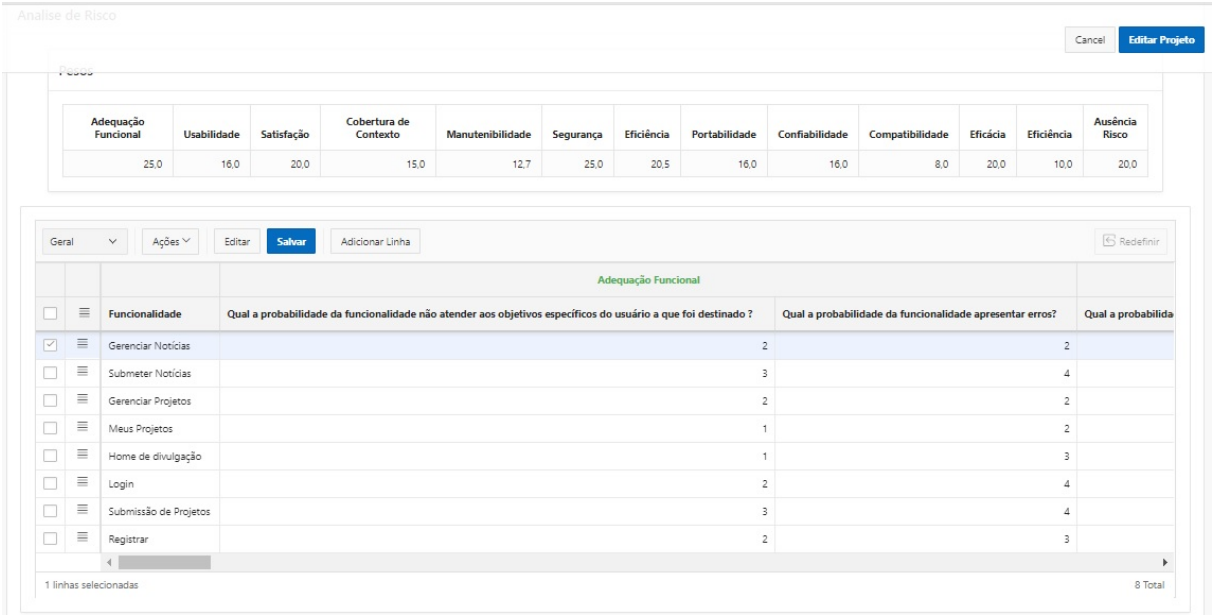



Figura 5.8: Análise de Risco - PUMA
Fonte: Elaborado pela autora

A fim de reduzir o tamanho da tabela as perguntas foram substituídas por número, entretanto a sua versão completa por ser vista pela ferramenta ou pela Figura 4.26, constante neste trabalho.

| Funcionalidades | FATORES DE DESENVOLVIMENTO (PROBABILIDADE) | | | | | | | | | | | | | | | | FATORES DE NEGÓCIO (IMPACTO) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Perguntas -> | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Gerenciar Notícias | 2 | 2 | 4 | 4 | 4 | 4 | 2 | 2 | 1 | 5 | 1 | 1 | 1 | 1 | 1 | 4 | 4 | 1 | 2 | 2 | 0 | 1 | 0 | 2 |
| Submeter Notícias | 3 | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 4 | 5 | 2 | 4 | 4 | 4 | 4 | 4 | 4 | 0 | 3 | 3 | 0 | 2 | 0 | 3 |
| Gerenciar Projetos | 2 | 2 | 4 | 4 | 2 | 4 | 2 | 2 | 1 | 5 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 4 | 0 | 3 | 0 | 2 | 0 | 3 |
| Meus Projetos | 1 | 2 | 4 | 4 | 2 | 4 | 2 | 5 | 1 | 5 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 3 | 3 | 0 | 3 | 0 | 2 |
| Home de divulgação | 1 | 3 | 4 | 4 | 2 | 2 | 2 | 2 | 5 | 1 | 1 | 1 | 3 | 1 | 2 | 2 | 2 | 4 | 1 | 4 | 0 | 4 | 0 | 3 |
| Login | 2 | 4 | 4 | 4 | 2 | 4 | 5 | 4 | 4 | 5 | 1 | 1 | 2 | 4 | 1 | 2 | 4 | 4 | 2 | 5 | 0 | 5 | 0 | 5 |
| Submissão de Projetos | 3 | 4 | 4 | 4 | 4 | 5 | 5 | 3 | 5 | 5 | 3 | 3 | 3 | 5 | 3 | 5 | 5 | 5 | 4 | 4 | 0 | 5 | 0 | 5 |
| Registrar | 2 | 3 | 4 | 3 | 2 | 4 | 5 | 1 | 5 | 5 | 2 | 3 | 2 | 4 | 3 | 2 | 3 | 4 | 3 | 5 | 0 | 5 | 0 | 5 |

Figura 5.9: Resultado da pontuação da Análise de Risco - PUMA
Fonte: Elaborado pela autora

A aplicação completa da fase de Identificação e Análise de Risco, foi realizada em 1hora e 48 minutos, compreendendo todos os passos citados nas seções 5.1.4 e 5.1.5. Uma vez concluída a fase de Análise de Risco junto a plataforma PUMA, a ferramenta apresentou a matriz de risco e o tratamento de risco, conforme apresentado nas seções 5.1.6 e 5.1.2.

### 5.1.6 Avaliação de Risco

Esta fase foi apresentada após a conclusão das fases citadas nas seções 5.1.4 e 5.1.5 por meio da disponibilização da matriz de risco, com o enquadramento dos itens de riscos (Funcionalidades/Requisitos) dispostos nos quadrantes da matriz.

A Figura 5.10, apresenta a matriz de risco e os Scores de Desenvolvimento e Negócio, gerados a partir da análise realizada na Plataforma Unificada de Metodologias Ativa - PUMA:


| Funcionalidades | Desenvolvimento | Negócio | Quadrante |
|---|---|---|---|
| Gerenciar Notícias | 45,06 | 26,00 | I |
| Gerenciar Projetos | 39,27 | 45,00 | I |
| Meus Projetos | 41,27 | 40,00 | I |
| Home de divulgação | 38,69 | 59,00 | I |
| Submeter Notícias | 67,53 | 35,00 | II |
| Registrar | 55,44 | 77,00 | III |
| Login | 58,57 | 75,00 | III |
| Submissão de Projetos | 73,09 | 79,00 | IV |

1 - 8

Figura 5.10: Matriz de Risco - PUMA
Fonte: Elaboradora pela autora

A matriz apresentada reflete o seguinte resultado:

- As funcionalidades Gerenciar Notícias, Gerenciar Projetos, Meus Projetos e Home de divulgação, foram dispostas no quadrante I, representando Baixo Impacto ao Negócio e Baixa Probabilidade de Ocorrência de falha.
- A funcionalidade Submeter Notícias, foi disposta no quadrante II, representando Baixo Impacto ao Negócio e Alta Probabilidade de ocorrência.

- As funcionalidades Registrar e Login, foram dispostas no quadrante III, representando Alto Impacto ao Negócio e Baixa Probabilidade.
- A funcionalidade de Submissão de Projetos, foi disposta no quadrante IV, representado Alto Impacto e Alta probabilidade de ocorrência.

O resultado foi apresentado e validado junto aos participantes do processo de avaliação de riscos para Qualidade de software. Com base na matriz apresentada, foi possível visualizar a priorização necessária que deveria compor o projeto em tempo de desenvolvimento. Uma vez concluído, foi possível visualizar o tratamento de risco, conforme apresentado na próxima na seção 5.1.7.

### 5.1.7 Tratamento do Risco

Essa fase foi executada após a conclusão do resultado apresentado pela matriz de risco conforme seção 5.1.6. Foi realizada uma análise junto a equipe técnica com a apresentação dos resultados providos pela ferramenta, com a estratégia completa de teste, conforme apresentado na Figura 5.11.

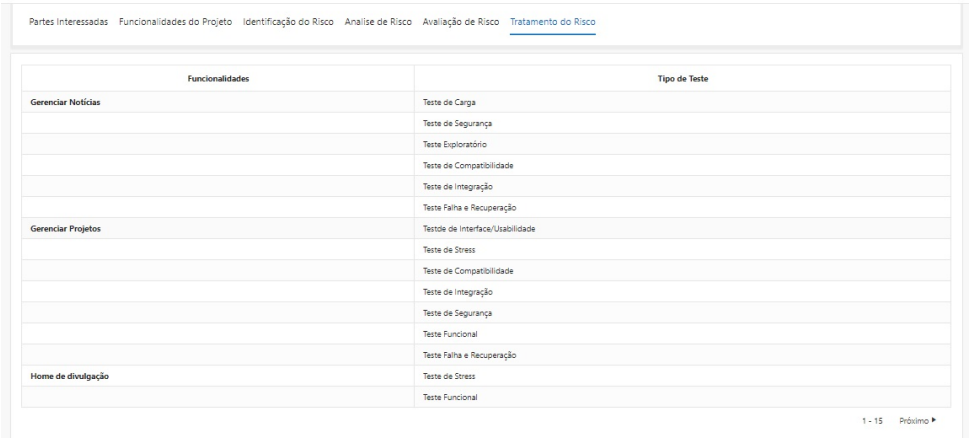



Figura 5.11: Tela do Tratamento do Risco - PUMA
Fonte: Elaborado pela autora

A Figura 5.12, apresenta a estratégia de teste completa para a plataforma PUMA, agrupados por funcionalidade.

| | |
|---|---|
| **Gerenciar Notícias** | Teste de Carga |
| | Teste de Segurança |
| | Teste Exploratório |
| | Teste de Compatibilidade |
| | Teste de Integração |
| | Teste Falha e Recuperação |
| **Gerenciar Projetos** | Testde de Interface/Usabilidade |
| | Teste de Compatibilidade |
| | Teste de Integração |
| | Teste de Segurança |
| | Teste Exploratório |
| | Teste Falha e Recuperação |
| **Home de divulgação** | Teste Exploratório |
| | Teste de Interface/Usabilidade |
| | Teste de Segurança |
| | Teste de Integração |
| | Teste de Performance |
| **Submeter Notícias** | Teste de Compatibilidade |
| | Teste de Carga |
| | Teste de Integração |
| | Teste de Segurança |
| | Teste Exploratório |
| | Teste Falha e Recuperação |
| **Submissão de Projetos** | Teste de Compatibilidade |
| | Teste Falha e Recuperação |
| | Teste de Integração |
| | Testde de Interface/Usabilidade |
| | Teste de Performance |
| | Teste de Segurança |
| | Teste Funcional |
| | Teste de Acessibilidade |
| | Teste de Carga |
| | Teste Funcional Automatizado |

| | |
|---|---|
| **Login** | Teste de Performance |
| | Teste de Compatibilidade |
| | Testde de Interface/Usabilidade |
| | Teste Funcional |
| | Teste de Acessibilidade |
| | Teste Falha e Recuperação |
| | Teste de Segurança |
| | Teste de Integração |
| | Teste de Carga |
| **Meus Projetos** | Teste de Integração |
| | Teste Exploratório |
| | Teste Falha e Recuperação |
| | Teste de Compatibilidade |
| | Teste de Segurança |
| | Testde de Interface/Usabilidade |
| **Registrar** | Teste de Integração |
| | Testde de Interface/Usabilidade |
| | Teste de Performance |
| | Teste Funcional Automatizado |
| | Teste Falha e Recuperação |
| | Teste Funcional |
| | Teste de Compatibilidade |
| | Teste de Acessibilidade |
| | Teste de Segurança |

Figura 5.12: Resultado Completo do Tratamento do Risco - PUMA
Fonte: Elaborado pela autora

A Figura 5.12, apresenta todos os tipos de teste sugerido pela ferramenta de Gestão de Risco para Qualidade de Software, para serem aplicados, com vistas a reduzir os riscos associados para cada funcionalidade.

Neste sentido, os tipos de testes sugeridos pela ferramenta, irão apoiar na redução dos riscos de falhas associados as características de qualidade, provendo um produto de mais qualidade. A Figura 5.13, apresenta quais características da qualidade, serão afetadas pelos testes.

| Tipos de Testes | Qual(is) Característica(s) de Qualidade da ISO 25010, este tipo de teste afeta? |
|---|---|
| Auditoria de código | Manutenabilidade |
| Teste de Acessibilidade | Usabilidade |
| Teste de Carga | Eficiência de desempenho |
| Teste de Compatibilidade | Compatibilidade, Portabilidade e Satisfação |
| Teste de Integração | Todos as características |
| Teste de Interface/Usabilidade | Usabilidade, Satisfação |
| Teste de Performance | Eficiência de desempenho |
| Teste de Segurança | Segurança |
| Teste Exploratório | Adequação Funcional, Eficácia |
| Teste Falha e Recuperação | Confiabilidade |
| Teste Funcional | Adequação Funcional, Eficiência, Ausência de Risco, Cobertura de Contexto |
| Teste Funcional Automatizado | Adequação Funcional |

Figura 5.13: Agrupamento dos tipos de testes para reduzir os riscos de falha nas características de qualidade - PUMA

Fonte: Elaborado pela autora

Em apresentação para equipe foi declarado que o único tipo de teste realizado inicialmente, foi o teste funcional onde é realizado na validação dos requisitos escritos em oposição ao que foi implementado. Apesar de entender ser uma grande quantidade de teste a ser realizado, foi declarado a necessidade da realização, uma vez que foram identificados vários erros, quando o software foi disponibilizado para teste beta, conforme apresentado na introdução deste estudo de caso. A avaliação dos resultados serão melhor explicados no Capítulo 6 deste trabalho.

### 5.1.8 Monitoramento e Análise Crítica

Esta fase iniciou-se após a implementação da estratégia de risco proposta pela ferramenta com vistas a assegurar e melhorar a qualidade e eficácia da implementação dos resultados do processo de gestão de risco da qualidade de software. Neste sentido, os painéis apresentam o status do andamento dos testes indicados por meio da tratamento de riscos. As Figuras 5.14 e 5.15 apresentam os resultados dos testes realizados mediante a estratégia apresentada pela ferramenta.

O Painel representado na Figura 5.14, apresenta todos os defeitos identificados nesta fase por gravidade e por status.

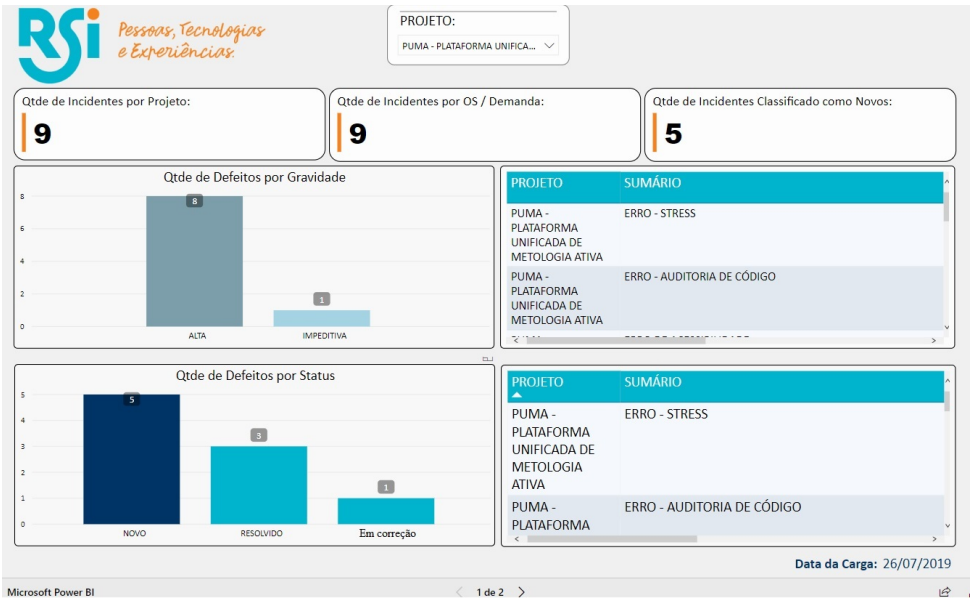



Figura 5.14: Painel de Indicadores de Erros - PUMA
Fonte: Elaborado pela autora

Já o painel representado na Figura 5.15, exibe todos os defeitos abertos por tipo de teste, por funcionalidade, por categoria de defeito e por prioridade.

Com base no tratamento de risco proposto, por meio da realização de teste, foi possível identificar uma série de defeitos. Alguns deles já conhecidos pela equipe, mais que poderiam ter sido evitados. Os erros identificados estão em correção, com expectativa de liberação em agosto com vistas a ser publicado em produção para captação de projetos a serem trabalhos no segundo semestre de 2019 na disciplina de Projeto de Sistemas de Produção 5. O Anexo I apresenta a tela da ferramenta mantis, com a evidência de todos os defeitos identificados, por meio da estratégia de teste indicada na ferramenta de Qualidade de Gestão de Risco para Software.

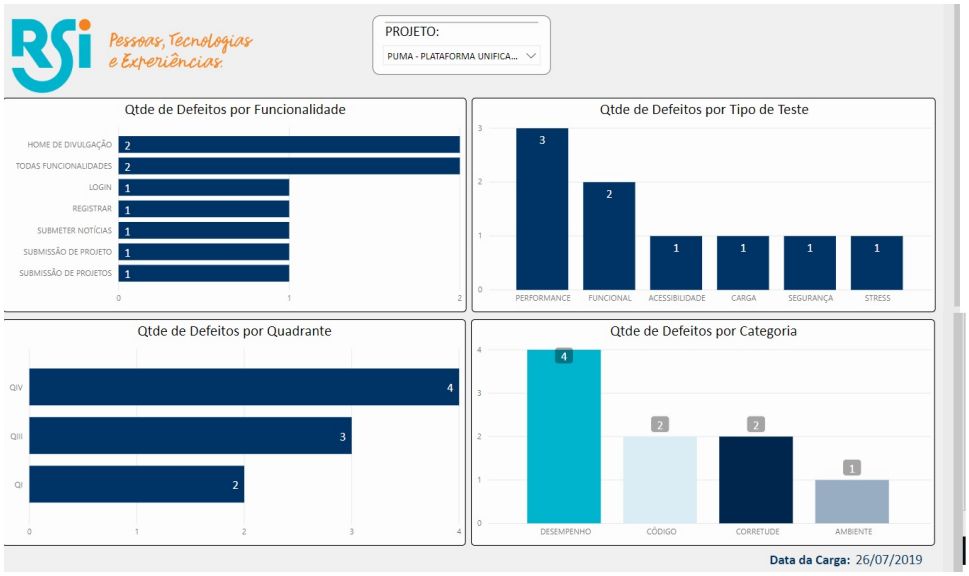



Figura 5.15: Painel de Indicadores de Erros - PUMA
Fonte: Elaborado pela autora

### 5.1.9 Registro e Reports

Esssa fase, foi composta pela documentação do processo de gestão de risco para Qualidade de software e seus resultados por meio de mecanismos apropriados. Para o projeto PUMA, foram realizados as seguintes atividades:

- Reunião com as partes interessadas para apresentação dos resultados do Processo de Gestão de Risco para de software;
- Apoio no embasamento de informações para tomadas de decisão; Documentação do processo por meio da ferramenta

A aplicação da ferramenta proposta na pesquisa, para este estudo de caso, permitiu a identificação de 17 riscos voltados ao produto, que afetam a 13 características de qualidade, possibilitando o mapeamento da probabilidade e impacto que cada evento de risco, pode afetar sobre o produto de software. Tendo como base os eventos de riscos identificados, foi possível realizar uma análise dos riscos, por meio das funcionalidades do produto, realizando a pontuação da probabilidade e impacto de ocorrência de falhas, dentre as características de qualidade da ISO25010. Uma vez concluída toda análise qualitativa, a ferramenta apresentou uma matriz de priorização, que permitiu uma visão de todas as funcionalidades críticas, permitindo assim um apoio na decisão de quais funcionalidades atacar primeiro em detrimento das com menos prioridades. Por fim, foram identificados a aplicação de 6 tipos de teste a serem aplicados no produto de software, com vistas as reduzir o risco de entrega com problemas de qualidade. Esses tipos de testes, permitiram

a identificação de 9 defeitos, sendo que 6 deles já tinham sido identificados na fase de pré-produção. Considerando o objetivo deste estudo de caso, na aplicação da ferramenta, com intuito de verificar se a ferramenta identificaria todos os defeitos já conhecidos pela equipe, a ferramenta se mostrou efetiva uma vez que mapeou mais defeitos do que os já identificados pela equipe.

## 5.2 Estudo de caso - PROJETO 2

### 5.2.1 Caracterização do Projeto

Este projeto, consiste no desenvolvimento de um sistema de consulta de processo para um órgão do governo federal. Foi desenvolvido pela fábrica de software contratada do cliente. Neste contexto a RSI Informática é a empresa contratada responsável pela prestação de serviços de Fábrica de Qualidade. O cliente aceitou realizar o estudo de caso aplicando a ferramenta de Gestão de Risco para Qualidade de software, entretanto foi solicitado a descaracterização das informações. Desta forma o cliente será nomeado como Cliente RSI e as partes interessadas serão nomeadas com outros nomes. Teve como escopo, a aplicação da ferramenta de Gestão de Risco para Qualidade software dentro da fases de desenvolvimento deste sistema. O projeto foi escolhido, uma vez que foi possível a obtenção dos resultados durante as fases de desenvolvimento e ainda o resultado final com a disponibilização do software em produção.

Trata-se de um sistema web, desenvolvido nas tecnologias: PHP 5.5 em banco de dados MySQL Server e servidor Apache Web Server - 2.4.6. Será utilizado em sua arquitetura a chamada a serviços já existes na busca de informações. A construção da aplicação ocorreu através da metodologia ágil por meio de 2 Sprints de 20 dias cada, conforme apresentado na Figura 5.16.

As funcionalidades compostas neste módulo, permitirá acesso ao público externo à consulta de processos, além dos acessos interno no módulo de administração para gerenciamento de perfis e níveis de acesso, bem como acesso a processos restritos internos ao órgão. Seguindo a estrutura proposta neste capítulo, segue as fases de aplicação da ferramenta para este estudo de caso.

### 5.2.2 Fase de Comunicação e Consulta

Essa fase, teve seu início por meio da reunião de *Sprint Planner*, junto o time de Scrum, *PO - Product Owner*, coordenador do projetos da TI e coordenador de teste. Na reunião foi definido o *backlog* das *Sprints*, conforme apresentado na Figura 5.16 e posteriormente foi apresentada a ferramenta, para a correta compreensão do objetivo da aplicação e

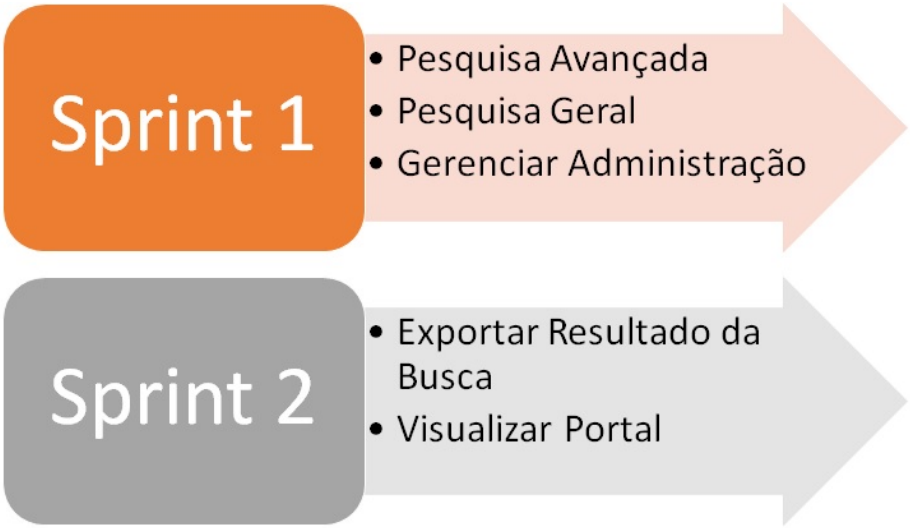



Figura 5.16: Planejamento das Sprints
Fonte: Elaborado pela autora

método estabelecido. Na reunião ficou definido também que a aplicação da ferramenta seria realizada por *Sprint* para a sua correta aplicação e coleta dos benefícios proposto. Foi possível também identificar as partes interessadas que irão participar do processo, conforme apresentada na Figura 5.17

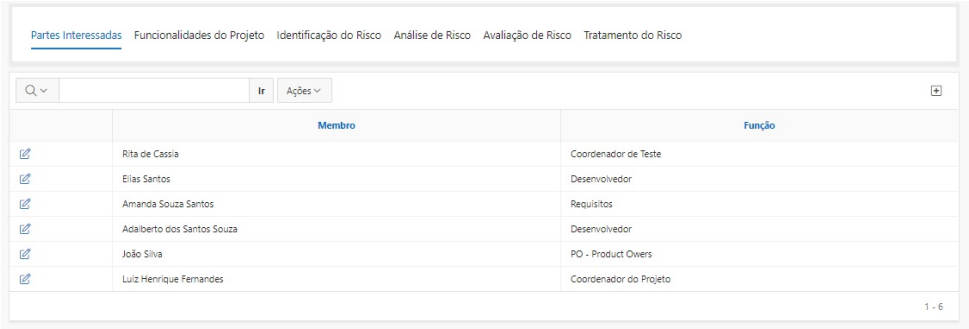



Figura 5.17: Fase de Comunicação e Consulta - Identificação das Partes Interessadas - PROJETO 2
Fonte: Elaboradora pela autora

A reunião ocorreu no início de maio e as entregas das *Sprints* foram definidas para entrega pela fábrica de software do Cliente RSI nos dias 29 de maio e 19 de junho.

### 5.2.3 Fase de Planejamento

Esta fase foi iniciada, através do estabelecimento do escopo para a aplicação da ferramenta de Gestão de Risco para Qualidade de software, por meio da identificação das informações gerais do produto em avaliação. Para esta fase, foram realizadas as seguintes atividades:

- Participação do gerente de teste, nas reuniões de *Sprints*, para a absorção do conhecimento do negócio;
- Avaliação do documento de visão do projeto;
- Avaliação das Especificações por Exemplos - EPES, com a descrição das funcionalidades;
- Avaliação do documento de arquitetura;
- Avaliação dos protótipos não funcionais;
- Confirmação das partes interessadas que irão participar do processo de Gestão de Risco para Qualidade de Software;
- Identificação das informações do produto, sendo eles: Segmento de Negócio, Público Alvo, Sigla do Projeto, Nome do Projeto, Responsável pelo Projeto e Organização;
- Identificação dos Itens de riscos que foram representados pelas funcionalidades/requisitos do software.

A Figura 5.18, apresenta todas informações levantadas para a fase de planejamento de acordo com as atividades acima listada.

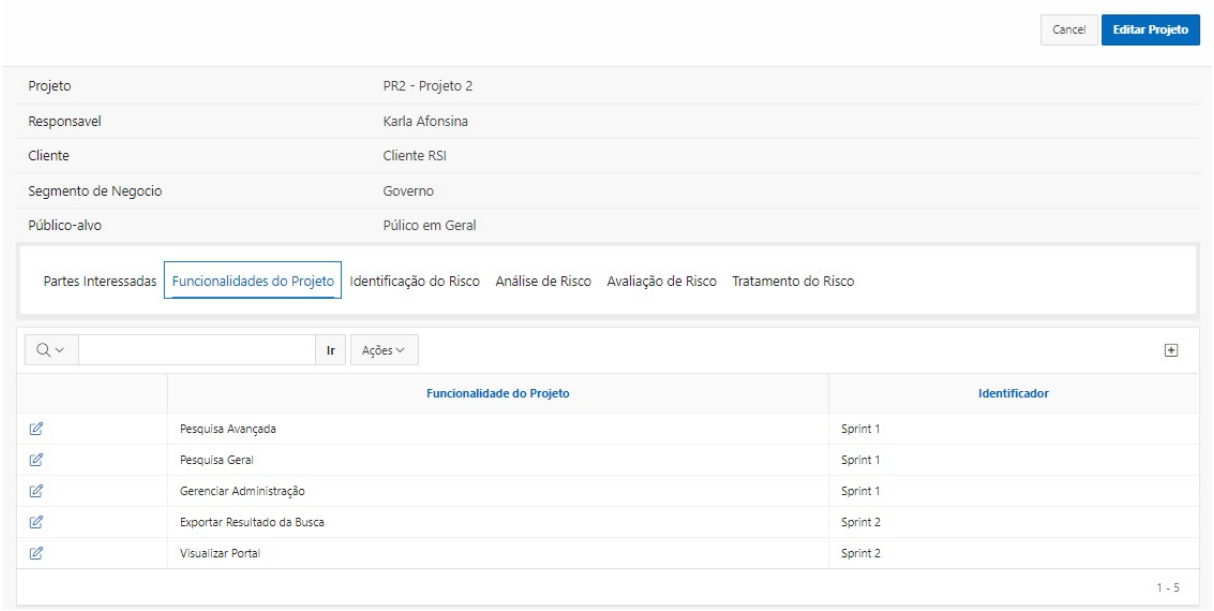



Figura 5.18: Fase de Planejamento - PROJETO 2
Fonte: Elaborado pela autora

As funcionalidades listadas na Figura 5.18 apresentada, foram mapeadas considerando o escopo de entrega das *Sprints*.

As próximas fases englobam o processo de avaliação de risco, sendo elas a Identificação de Riscos, Análise de Riscos e Avaliação dos Riscos, conforme detalhamento a seguir.

### 5.2.4 Identificação de Risco

Essa fase ocorreu durante 2 momentos, sendo o primeiro na construção da *Sprint* 1 e o segundo no desenvolvimento da *Sprint* 2. Considerando o modelo de desenvolvimento de software deste projeto, foi realizada uma reunião com o time de SCRUM e o Product Owner (PO) com objetivo de mapear os riscos que poderiam afetar a qualidade do produto do ponto de vista de desenvolvimento e negócio. Foi realizado por meio de uma reunião específica para este assunto, e liderada pela autora deste trabalho e gerente de teste da empresa RSI Informática. Na oportunidade, foi possível mapear as seguintes informações nessa fase:

- Mapeamento dos eventos de riscos, que podem afetar a qualidade do produto de software;
- Categorização dos riscos identificados, por meio das características de qualidade;
- Classificação da probabilidade de ocorrência, por meio da pontuação dos riscos, seguindo a matriz de definição de probabilidade, conforme Figura 4.10.

A Figura 5.19, apresenta todos os riscos identificados pela time Scrum e PO do projeto:

Os riscos identificados e qualificados foram em sua grande maioria mapeados na *Sprint 1*. Na *Sprint 2* foram revisitados, sendo necessário a inclusão de mais um item. A seguir são listado todos os riscos identificados nesta fase.

- A não usabilidade nas funcionalidades públicas foi classificada como um risco, vez que público alvo do sistema é, em sua grande maioria, formado de advogados, que têm uma visão crítica mais apurada;
- A realização de manutenção em uma funcionalidade e o provável impacto em outras funcionalidades foi identificado como um risco por dois pontos de vista: Técnico, já que a arquitetura do órgão ainda é monolítica e do ponto de vista de negócio, pois a experiência do Product Owner (PO) proveniente de outros projetos que tem como histórico erros em outras partes dos sistema após a manutenção do software. A reunião ainda permitiu discussões acerca da arquitetura do órgão, ficando evidente a necessidade de melhoria;
- A insatisfação dos usuários externos, dado que a versão que será entregue possa não atender em sua plenitude toda necessidade. Além disso, o público alvo tende a ser mais criterioso;

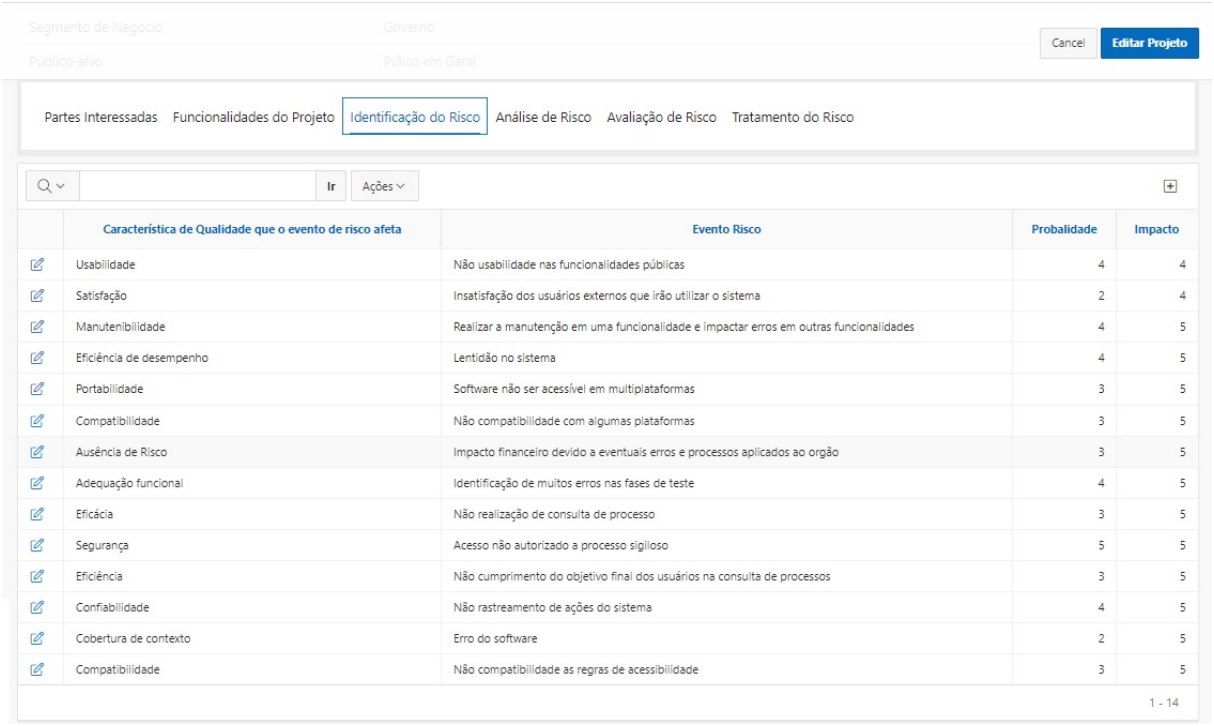



Figura 5.19: Riscos Identificados - PROJETO 2
Fonte: Elaboradora pela autora

- O fato de o software não ser acessível em multiplataformas foi apontado como um risco, já que atualmente plataformas são variadas as múltiplas. Conseguir abarcar a sua maioria não é uma tarefa simples;
- O acesso não autorizado a processo sigiloso foi mapeado como risco, pois esses só podem ser acessados por pessoas específicas. O acesso às informações de processos sigilosos pode apresentar impactos imensuráveis ao negócio, tanto do ponto de vista de imagem quanto ao ponto de vista financeiro;
- Lentidão no sistema foi identificada como um risco. É de suma importância que o programa tenha bom desempenho principalmente nas funcionalidades externas de consulta de processo. A lentidão causa impactos de imagem ao órgão, principalmente em dias de alta sazonalidade, como é o caso de audiências;
- O não cumprimento do objetivo final dos usuários na consulta de processos, foi mape- ado como risco posto que a consulta de processos é a finalidade do sistema. Caso isso não ocorra pode haver impactos quanto à a visibilidade no negócio;
- O não rastreamento de ações do sistema, foi identificado como um risco. Os dados disponibilizados para consulta são sensíveis e investigar as ações de usuários é de suma importância para possíveis rastreabilidades de ações não desejadas;

- Defeito do software foi considerando como um risco, mesmo que baixo, pois está sendo aplicado à ferramenta desta pesquisa;
- Impacto financeiro devido a eventuais defeitos foi mapeado como um risco devido ao histórico dos gestores. Eventuais impedimentos no acesso ao software ou apresentação de informações sigilosas indevidamente, podem desdobrar em processos ao órgão;
- A identificação de muitos erros nas fases de teste foi mapeado como um risco, vez que a fábrica de software tem histórico de entregas ruins e de má qualidade;
- A não compatibilidade com as regras de acessibilidade foi listado com um risco, já que será aberto ao público externo a pesquisa pública e o portal. Assim é de suma importante que as regras de acessibilidade sejam aplicadas para que não haja restrição de o acesso ao software;
- A não realização de consulta a processos também foi mapeada como uma risco, pois é o principal objetivo do sistema desenvolvido. A não realização do objetivo principal impacta a característica de qualidade e eficácia.

Para este projeto, foi necessário que no momento da identificação do risco, uma explicação das características de qualidade de maneira mais detalhada fosse exposta com o objetivo de explorar mais as possibilidades de possíveis riscos. Outro ponto a ser destacado foi a participação da gerente de teste nas reuniões diárias de *daily* e de levantamento de requisitos ao decorrer das *Sprints*. Este fato levou a um melhor entendimento do negócio. Após esse mapeamento, a próxima fase realizada foi a de Análise de riscos, conforme detalhamento a seguir.

### 5.2.5 Análise de Risco

Essa fase, teve continuidade na mesma reunião detalhada na fase de identificação de risco. O principal objetivo foi a análise qualitativa dos itens que são representados pelas funcionalidades junto as perguntas das características de qualidade previstas na ferramenta. Essa fase, foi seguida das seguintes atividades:

- Listagem dos itens de riscos, que são apresentados todos já cadastrados na aba de "Funcionalidades do Projeto";
- Pontuação das perguntas relacionadas aos Fatores de Desenvolvimento (Probabilidade), realizada junto ao time técnico, considerando a matriz de definição de probabilidade 4.10;

- Pontuação das perguntas relacionadas ao Fatores de Negócio (Impacto) junto ao responsável pelo projeto e Product Owner (PO), considerando a matriz de definição de probabilidade 4.10;
- Consenso das pontuações apresentadas, também durante a reunião.

A Figura 5.20, apresenta a tela da ferramenta com a Análise de risco realizada.

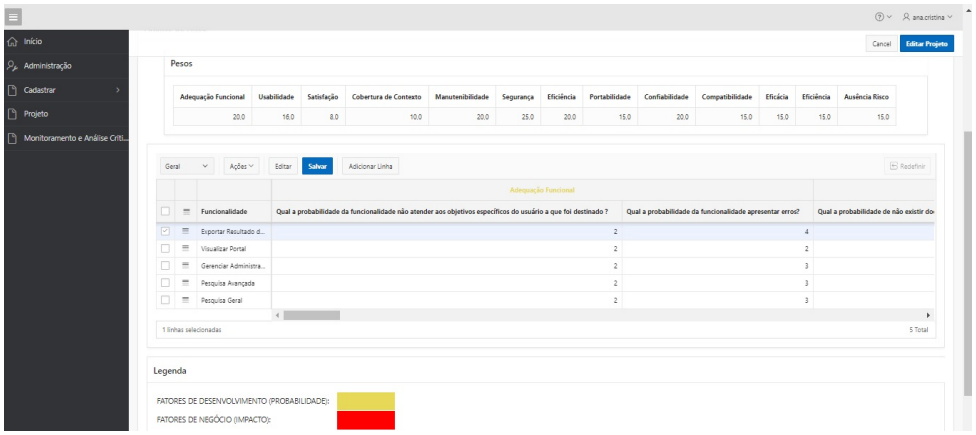



Figura 5.20: Análise de Risco - PROJETO 2
Fonte: Elaborado pela autora

A Figura 5.21 exibe o resultado completo da pontuação realizada pela equipe técnica e de negócio para o PROJETO 2.

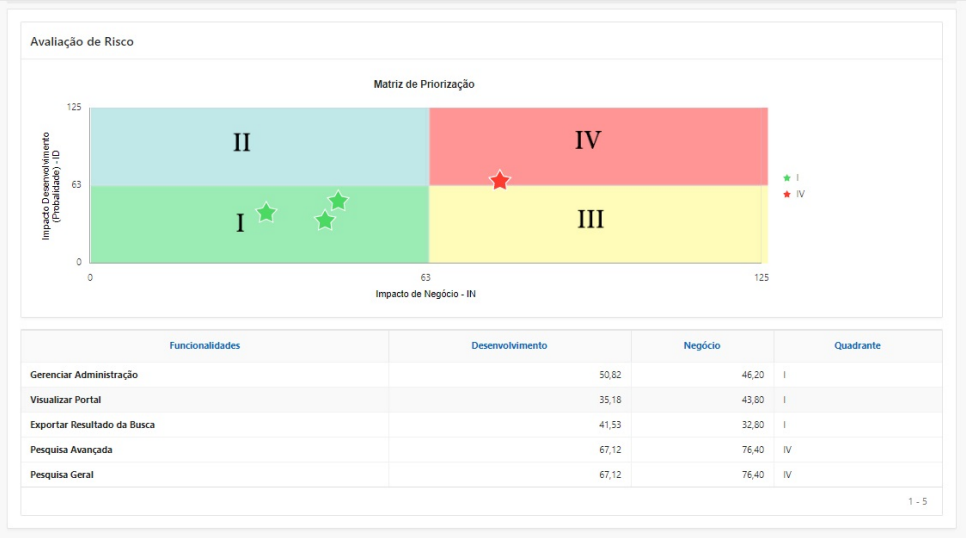
| Funcionalidades | FATORES DE DESENVOLVIMENTO (PROBABILIDADE) | | | | | | | | | | | | | | | | | FATORES DE NEGÓCIO (IMPACTO) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Perguntas -> | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Exportar Resultado da Busca | 2 | 4 | 1 | 1 | 2 | 1 | 4 | 1 | 4 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 4 | 2 | 3 | 3 | 0 | 2 | 1 | 2 |
| Visualizar Portal | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 3 | 2 | 2 | 1 | 1 | 4 | 4 | 2 | 2 | 4 | 3 | 3 | 0 | 3 | 1 | 3 |
| Gerenciar Administração | 2 | 3 | 1 | 4 | 4 | 4 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 4 | 4 | 2 | 0 | 4 | 1 | 2 |
| Pesquisa Avançada | 2 | 3 | 1 | 4 | 4 | 4 | 4 | 5 | 4 | 3 | 2 | 3 | 2 | 4 | 4 | 4 | 5 | 5 | 5 | 4 | 0 | 5 | 5 | 5 |
| Pesquisa Geral | 2 | 3 | 1 | 4 | 4 | 4 | 4 | 5 | 4 | 3 | 2 | 3 | 2 | 4 | 4 | 4 | 5 | 5 | 5 | 4 | 0 | 5 | 5 | 5 |

Figura 5.21: Resultado da pontuação da Análise de Risco - PROJETO 2
Fonte: Elaborado pela autora

A aplicação completa da fase de Identificação e Análise de Risco, foi realizada em 1hora e 24 minutos contemplando as 2 *Sprints*, compreendendo todos os passos citados nas seções 5.2.4 e 5.2.6. Uma vez concluída a fase de Análise de Risco junto ao PROJETO 2 a ferramenta apresentou a matriz de risco e a tratamento de risco, conforme apresentado nas duas próximas seções.

### 5.2.6 Avaliação de Risco

Essa fase foi representada por meio da matriz de risco, com enquadramento dos itens de riscos (Funcionalidades/Requisitos), dispostos nos quadrantes da matriz conforme apresentado na Figura 5.22 e ainda o Score de Desenvolvimento e Negócio, calculadas a partir das pontuações realizadas nas abas de Identificação e Análise de Risco.




| Funcionalidades | Desenvolvimento | Negócio | Quadrante |
|---|---|---|---|
| Gerenciar Administração | 50,82 | 46,20 | I |
| Visualizar Portal | 35,18 | 43,80 | I |
| Exportar Resultado da Busca | 41,53 | 32,80 | I |
| Pesquisa Avançada | 67,12 | 76,40 | IV |
| Pesquisa Geral | 67,12 | 76,40 | IV |

1 - 5

Figura 5.22: Matriz de Risco - PROJETO 2
Fonte: Elaborado pela autora

A matriz gerada para o PROJETO 2, representa as seguintes informações:

- As funcionalidades Gerenciar Administração, Visualizar Portal e Exportar Resultado da Busca, foram dispostas no quadrante I, representando Baixo Impacto ao Negócio e Baixa Probabilidade de Ocorrência de falha.
- As funcionalidades Pesquisa Avançada e Pesquisa Geral, foram disposta no quadrante IV, representando Alto Impacto e Alta probabilidade de ocorrência. Neste caso, ambas as funcionalidades apresentaram exatamente o mesmo Score de Desenvolvimento e Negócio, sendo exibida na matriz uma sobreposta a outra. Este fato ocorreu porque as funcionalidades são praticamente as mesmas, mudando apenas a forma de pesquisa para cada uma.

O resultado da matriz foi discutida pela equipe participante da avaliação, e representou exatamente a mesma percepção que já se tinha dentro do projeto. Inclusive foi percebido que a priorização para a construção das funcionalidades, seguiram essa mesma estrutura. Apresentada a matriz de risco, foi possível visualizar o tratamento de risco sugerido pela ferramenta com a finalidade de reduzir a probabilidade e impactos das ocorrências como que será apresentado na próxima seção.

### 5.2.7 Tratamento do Risco

Essa fase foi realizada após a conclusão das fases descritas nas seções anteriores. Foi realizada uma análise junto a equipe que participou do processo de avaliação, onde foram apresentados os resultados originários pela ferramenta de Gestão de Risco para Qualidade

de software. A Figura 5.23, apresenta a tela com os resultados gerados da estratégia de teste gerada para a ferramenta e está agrupada por funcionalidade.

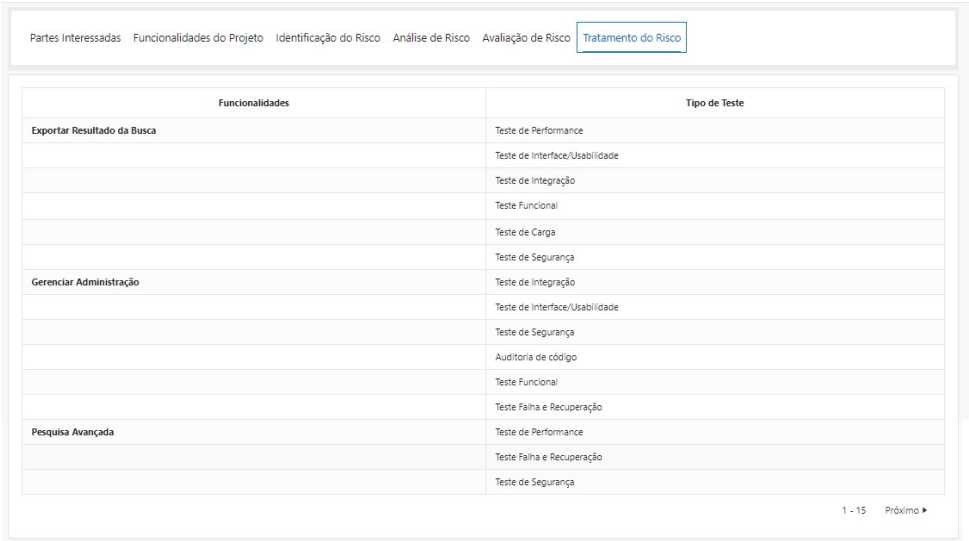



Figura 5.23: Tela do Tratamento dos Riscos - PROJETO 2
Fonte: Elaborado pela autora

A Figura 5.24, apresenta todos os tipos de testes sugerido pela ferramenta, a serem aplicados nas funcionalidades, com a finalidade de reduzir os riscos de uma entrega com falhas, afetando assim a qualidade do produto de software entregue.

| Funcionalidades | Tipo de Teste |
|---|---|
| **Exportar Resultado da Busca** | Teste de Performance |
| | Teste de Interface/Usabilidade |
| | Teste de Integração |
| | Teste Funcional |
| | Teste de Carga |
| | Teste de Segurança |
| **Gerenciar Administração** | Teste de Integração |
| | Teste de Interface/Usabilidade |
| | Teste de Segurança |
| | Auditoria de código |
| | Teste Funcional |
| | Teste Falha e Recuperação |

| Funcionalidades | Tipo de Teste |
|---|---|
| **Pesquisa Avançada e Pesquisa Geral** | Teste de Performance |
| | Teste Falha e Recuperação |
| | Teste de Segurança |
| | Teste de Interface/Usabilidade |
| | Teste de Carga |
| | Teste de Compatibilidade |
| | Teste Funcional |
| | Teste de Acessibilidade |
| | Teste de Integração |
| | Teste Funcional Automatizado |
| | Auditoria de código |
| **Visualizar Portal** | Teste de Interface/Usabilidade |
| | Teste de Integração |
| | Teste de Performance |
| | Teste Funcional |
| | Teste de Acessibilidade |

Figura 5.24: Resultado Completo do Tratamento dos Riscos - PROJETO 2
Fonte: Elaborado pela autora

Essa análise deu subsídio para a elaboração da estratégia de teste a ser realizada nas *Sprints 1 e 2* do projeto. A dimensão da quantidade de teste foi alta, pois o time, foi identificou a necessidade para o contexto apresentado.

| Tipos de Testes | Qual(is) Característica(s) de Qualidade da ISO 25010, este tipo de teste afeta? |
|---|---|
| Auditoria de código | Manutenabilidade |
| Teste de Acessibilidade | Usabilidade |
| Teste de Carga | Eficiência de desempenho |
| Teste de Compatibilidade | Compatibilidade, Portabilidade e Satisfação |
| Teste de Integração | Todos as características |
| Teste de Interface/Usabilidade | Usabilidade, Satisfação |
| Teste de Performance | Eficiência de desempenho |
| Teste de Segurança | Segurança |
| Teste Falha e Recuperação | Confiabilidade |
| Teste Funcional | Adequação Funcional, Eficiência, Ausência de Risco, Cobertura de Contexto |
| Teste Funcional Automatizado | Adequação Funcional |

Figura 5.25: Agrupamento dos tipos de testes para reduzir os riscos de falhas nas características de qualidade - PROJETO 2
Fonte: Elaborado pela autora

A Figura 5.25, apresenta um agrupamento de todos os tipos de testes necessários, para reduzir os riscos de falhas, associados as características de qualidade. Uma vez apresentado o tratamento de riscos, a próxima seção exibe, como será realizado o Monitoramento e Análise Crítica, dos tipos de testes sugeridos, com vistas a assegurar que todo o tratamento de risco seja aplicado.

### 5.2.8 Monitoramento e Análise Crítica

Esta fase foi iniciada, após a implementação da estratégia de risco proposta pela ferramenta. Neste cenário, foi permitido um acompanhamento através dos painéis apresentados a seguir. A Figura II.1, exibe o painel dos indicadores de defeitos por gravidade e defeitos por status.

Já o 5.27, apresenta o andamento dos testes, o status das correções dos defeitos identificados e da quantidade de defeitos por tipo de teste .

Com base no tratamento de risco proposto, por meio da realização de teste foi possível identificar uma série de defeitos. O gestor da TI solicitou que todos os erros identificados fossem corrigidos, sendo que a premissa para produção e disponibilização aos usuários era essa correção.

O Anexo II apresenta a tela da ferramenta Mantis, com a evidência de todos os defeitos identificados, por meio da estratégia de teste indicada na ferramenta de Qualidade de Gestão de Risco para Software.

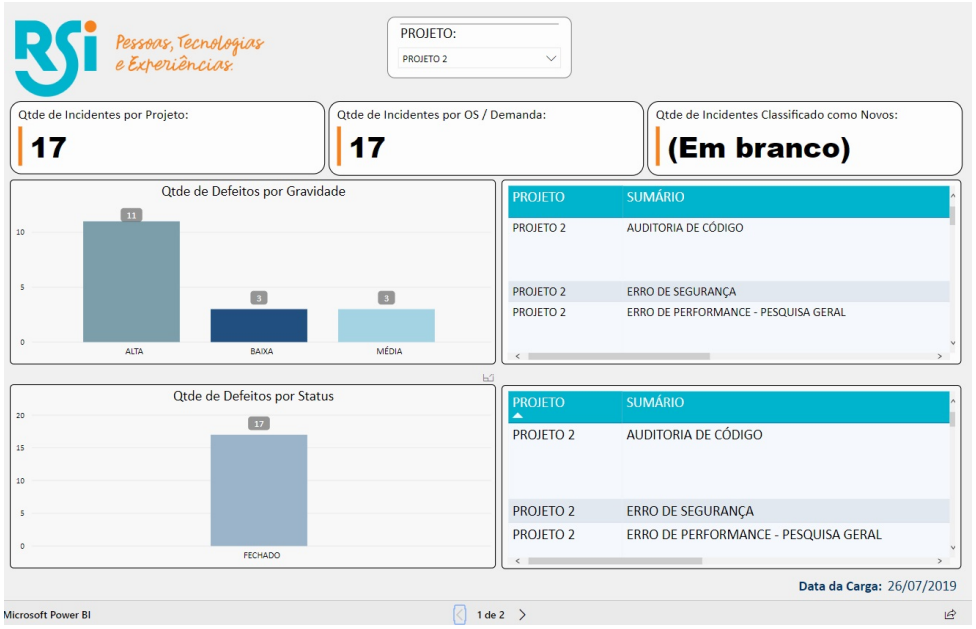



Figura 5.26: Painel 1 de Indicadores de Erros - PROJETO 2
Fonte: Elaborado pela autora

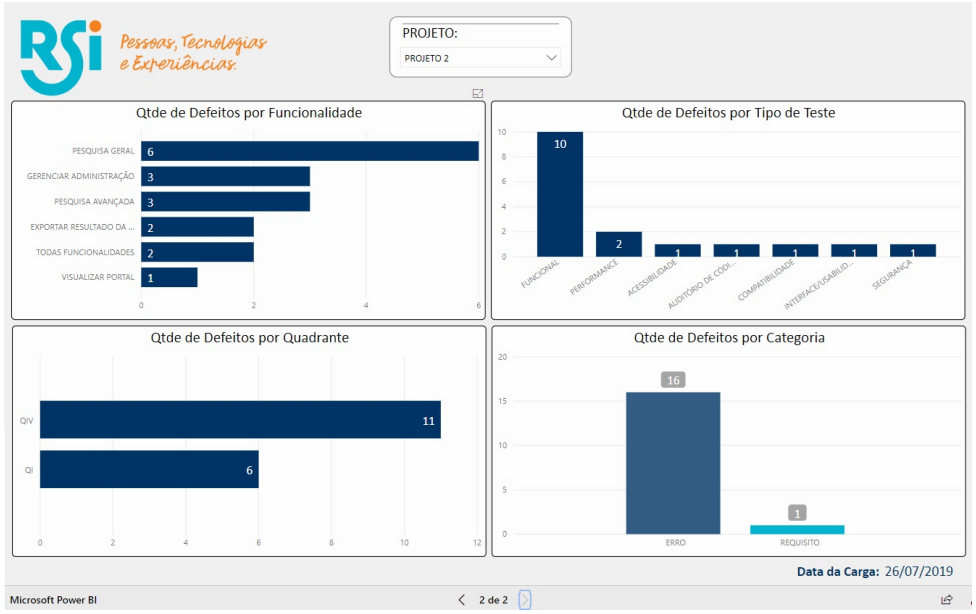



Figura 5.27: Painel 2 de Indicadores de Erros - PROJETO 2
Fonte: Elaborado pela autora

### 5.2.9 Registro e Reports

Essa fase foi composta pela documentação do processo de gestão de risco para Qualidade de Software e seus resultados através de mecanismos apropriados. Para o PROJETO 2 foram realizadas as seguintes atividades:

- Reunião com as partes interessadas para apresentação dos resultados do Processo de Gestão de Risco para de software;
- Apoio no embasamento de informações para tomadas de decisão; Documentação do processo por meio da ferramenta.

O estudo de caso resultou na identificação de 15 eventos de riscos relacionados às 13 características de qualidade e permitiu, por meio da análise de risco, a visibilidade de 2 funcionalidades críticas, mediante a matriz de priorização, que norteou a equipe em focar nas funcionalidades mais importantes. A equipe do projeto, reportou que, sem os resultados da ferramenta não seria possível, ainda em tempo de desenvolvimento a identificação dos defeitos de performance, acessibilidade, auditória de código, compatibilidade, interface e segurança contabilizando 7 problemas, já que o processo de teste interno do órgão, estabelece apenas a realização de teste funcional. Ao final desse processo foram identificados um total de 16 defeitos, que passaram por correção, antes da disponibilização em seu ambiente produtivo. A validação da ferramenta por meio do estudo de cado foi efetiva, pois foi possível a realização de testes dentro da fase de desenvolvimento de software através de uma estratégia definida pela ferramenta e por meio da gestão de risco que apoiou na obtenção de maior qualidade do software, uma vez que identificou antecipadamente 7 defeitos que inicialmente pela estratégia definida pelo órgão, são seria identificado, permitindo assim uma entrega de qualidade.

Apresentados os estudos de caso, deu-se por concluída a aplicação da ferramenta de Gestão de Risco para Qualidade de software. O Capítulo 6 apresenta uma avaliação dos resultados das aplicações.

# Capítulo 6

# Avaliação dos Resultados da Aplicação da Ferramenta

Este capítulo tem como objetivo apresentar os resultados da validação da ferramenta de Gestão de Risco para Qualidade de software, descrevendo suas principais constatações quanto às suas características, aplicabilidade e resultados; em seguida apresenta como a ferramenta será implantada na RSI informática e os principais benefícios obtidos com a sua adoção.

## 6.1 Validação da Ferramenta de Gestão de Risco para Qualidade de software

Essa seção apresenta a análise qualitativa dos resultados da validação da ferramenta de gestão de risco para Qualidade de software, detalhando os resultados para cada fase de aplicação da ferramenta e os métodos utilizados.

A Fase de Comunicação e Consulta possibilitou a identificação das partes chave interessadas nos produtos avaliados e contribuiu para o aumento do conhecimento dos envolvidos em relação às informações do processo de aplicação da ferramenta. Além disso facilitou a execução das demais atividades das fases de aplicação e de utilização da ferramenta ao fornecer informações necessárias para apoiar sua realização.

A Fase de Planejamento, possibilitou o entendimento do escopo dos produtos a serem aplicados na ferramenta de Gestão de Risco para Qualidade de Software, os ativos de processos organizacionais e os itens de riscos por meio da documentação analisada dos produtos. Essa fase foi suma importância, pois permitiu o a comprrensão negocial do produto, para apoio na aplicação da ferramenta e do processo. O projeto 2, destacou-se na realização desta fase, uma vez que a participação nas reuniões de *Sprints* e *Daily*,

permitiu um conhecimento negocial mais intenso do sistema, apoiando a equipe de teste na identificação dos riscos e sua devida análise qualitativa.

Em seguida a fase de identificação dos riscos, permitiu explorar os riscos dos produtos na identificação das incertezas, que podem afetar a qualidade do software. Como resultado dos estudos de caso, foram possíveis a identificação de 17 riscos no projeto PUMA e 14 riscos no PROJETO 2. Para o PROJETO 2 os riscos mapeados, permitiram gerar uma análise crítica ainda na fase de desenvolvimento, permitindo assim uma melhor condução do projeto com uma entrega de mais qualidade. Para o Plataforma Unificada de Metodologia Ativa (PUMA), a identificação dos riscos, refletiu a necessidade de alguns ajustes na forma de desenvolvimento a serem aplicados no módulo 2 do projeto, com vistas a já mitigar alguns dos riscos elencados no módulo 1.

A execução da fase de Análise de Riscos proporcionou um estudo qualitativo dos itens de risco do produto em avaliação. Neste sentido, para cada funcionalidade foi associada uma pontuação de probabilidade e impacto de ocorrência de falha considerando todas as características de qualidade previstas na ISO[2], e apresentadas na Figura 4.26.

Para o PROJETO 2, as perguntas permitiram reflexão na Metodologia de Desenvolvimento de Software atualmente estruturada no órgão, deixando claro a necessidade de melhoria no processo interno, conforme listadas:

- Necessidade de mudança da arquitetura monolítica para arquitetura voltada a micro serviços a fim de mitigar alguns riscos de manutenibilidade;
- Aplicação de práticas de desenvolvimento seguro, já incorporados ao processo;
- Criação de uma guia de Interface e Acessibilidade;
- Melhoria na montagem dos ambientes com dados mais expressivos;
- Melhoria na Metodologia de Desenvolvimento de Software, vez que em alguns momentos foi questionado pela fábrica a não normatização de padrões.

Já a fase de Avaliação de Riscos, propiciou por meio da matriz de risco um apoio a tomada de decisão referente a priorização das funcionalidades por meio dos quadrantes I, II, III e IV. A priorização em relação a execução dos tipos de testes apresentados por meio da funcionalidade de tratamento de risco, ocorreu seguindo o método apresentando, considerando os itens a seguir:

- Foi iniciada a atuação por meio das funcionalidades apresentadas no quadrante 4, que representa alta probabilidade de ocorrência e alto impacto ao negócio caso ocorra;

- Posterioriormente, foi atacado as funcionalidades do quadrante III, que apesar de ter baixa probabilidade de ocorrência, caso ocorra falha o impacto é alto;
- Em sequência, foi atuado no quadrante II que apresenta alta probabilidade de ocorrência de falha apesar do impacto ser menor;
- E por último, para as funcionalidades do quadrante I que representam baixo impacto e baixa probabilidade foram concedido menos prioridade.

Ao perguntar aos participantes do processo se a matriz gerada corresponde a realidade do negócio, foi que sim para ambos os projetos e que agrega valor na tomada de decisão apoiando na priorização e onde alocar mais esforços. Com relação ao PROJETO 2 a matriz refletiu exatamente a percepção dos envolvidos que informa ainda que nos momentos de necessidade de curto prazo, será de grande serventia a matriz gerada não só ao produto de software, mais também ao projeto como um todo.

A Fase de Tratamento de Riscos, apoiou na implementação da estratégia de teste com a finalidade de reduzir a má qualidade do produto de software. A ferramenta abordou uma série de testes a serem realizados para reduzir a probabilidade de ocorrência de falhas. Inicialmente para os estudos de caso apresentados, sem a visibilidade da ferramenta e do método, foi proposto como estratégia apenas a realização de teste funcional.

Para o PUMA, ao expor todos os tipos de teste indicados, conforme apresentado na seção 5.1.7, já foi percebido que alguns erros identificados na pré-produção poderiam ter sido evitados, com a estratégia de teste apresentada na ferramenta, conforme listados abaixo:

- Foi identificado no ambiente de pré-produção problemas de lentidão e indisponibilidade. Ao ter apenas 3 usuários simultâneos a aplicação apresentava lentidão e posteriormente ficava indisponível. Estes erros foram identificados principalmente nas funcionalidades Home de Divulgação, Login, Registrar e Submissão de Projetos. Para essas funcionalidades a ferramenta sugeriu a realização de teste de performance. Para o login, foi sugerido também o teste de *stress* que também permitiria a identificação dos problemas apontados. As funcionalidades elencadas basicamente trabalham com *uploads* de arquivos e este fato exige uma performance mais robusta, com exceção do login que por sua vez precisa suportar uma quantidade de pelo menos 100 usuários simultâneos, de acordo com os requisitos não funcionais do PUMA.
- Foi listado teste de segurança para todas as funcionalidades. Em alinhamento com a equipe, foi identificado que essa necessidade se deve ao fato, que a plataforma não foi desenvolvida seguindo as melhoras práticas de desenvolvimento seguro. Isso

decorre de mais um risco que foi mapeado posterior a execução da fase de Análise de Risco, que é a hospedagem no Centro de Processamento de Dados (CPD) devido às regras de segurança aplicada no departamento. Com a realização de testes de segurança, foram identificados 27 vulnerabilidades, conforme apresentado na seção de monitoramento e Análise Crítica. Como ação adicional a equipe de desenvolvimento irá aplicar uma refatoração no módulo I e inclusão das práticas para o módulo II.

- Foi sugerido também uma auditoria de código. Esse foi realizado posterior a liberação de pré-produção e não foram identificados erros no código, entretanto foram apontados alguns alertas de 4.2% de código duplicado e 33 inconformidade, referente ao códido fonte, fora dos padrões.

- Também na realização do teste beta, algumas submissões de projetos apresentaram erros e a aplicação se comportou de maneira inesperada apresentando um erro para o usuário e não recuperando a informações já reportadas na plataforma. Neste sentido a ferramenta sugeriu um teste de falha de recuperação.

- Foi identificado no ambiente de pré-produção problemas de compatibilidade e interface. Ao acessar por dispositivos móveis a aplicação não se comporta de maneira correta, inclusive na funcionalidade de submissão de projetos. Nesse sentido alguns projetos não foram submetidos, devido a estes erros, causando impactos. Os testes de compatibilidade e interface de usuário foram sugeridos pela ferramenta. Foi declarado pela equipe, que apesar da tecnologia utilizada já ajudar nessa adaptação a plataforma não está preparada para dispositivos mobiles e algumas versões do *browser* como Mozilla e *IE - Internet Explorer*,

Para as demais funcionalidades, percebe-se as seguintes estratégias:

- Para as funcionalidades Gerenciar Notícias, Gerenciar Projetos, Home de Divulgação, Submeter Notícias e Meus Projetos foi indicado fazer teste exploratório, teste este mais simples, ao qual se navega pela tela explorando possíveis problemas. A equipe relatou que esta estratégia não foi pensada inicialmente e foi realizado testes funcionais para todas as funcionalidades, gastando assim, mais tempo em sua realização que poderiam ser utilizados nos outros tipos de teste sugeridos.

- Já para as funcionalidades de Submissão de Projetos, Registrar, Login e Submissão de Projetos foi indicado fazer um teste funcional, que é mais detalhado, justamente por essas serem as principais funcionalidades utilizadas pelo público externo.

- Para a funcionalidade de Submissão de Projeto, foi indicado a realização de testes automatizados, que simulas de forma automatizada o teste funcional, uma vez que a

funcionalidade foi pontuada com uma maior probabilidade de ocorrência, estando na escala de Muito Alta, ou seja, mais de 80% de probabilidade de ocorrência de falha. O teste automatizado, neste sentido, irá rodar automaticamente sem a necessidade de ação humana e caso seja identificado problemas, não permitirá a disponibilização para o ambiente de produção com a falha indicada. Foi percebido que este tipo de teste apoiará na redução dos esforços de testes manuais, uma vez que toda release disponibilizada, passará pelo teste automatizado e caso tenha algum defeito, será exposto. Entretanto mesmo entendendo o benefício deste tipo de teste, foi definido pela equipe a não realização neste momento.

- Foi indicado, teste de Interface/Usabilidade para as funcionalidades de Submissão de Projetos, Gerenciar Projetos, Login, Registrar e Meus Projetos. Neste cenário a equipe relata também ser a realidade do projeto, dado que todas as funcionalidades listadas são utilizadas pelo público externo e a usabilidade se torna crítica para a utilização do software. Para esse teste foi identificado 86% de aderência de 20 ocorrências.

- Foi elencado teste de falha e recuperação para as funcionalidades: Gerenciar Notícias e Projetos, Submeter Notícias, Submissão de Projetos, Login, Meus Projetos e Registrar. Este teste identifica o quão o software está preparado para recuperar os dados, em caso de falha. A equipe entende que com exceção das funcionalidades Gerenciar Notícias, Gerenciar Projetos e Meus Projetos, não faz sentido este tipo de teste, pois é basicamente consulta e não tem inserção de dados. Para os demais, faz-se necessário a realização desses testes, que inclusive foi identificado que aplicação não está conforme.

Para o PROJETO 2, os tipos de testes apresentados no tratamento de riscos foi implementado dentro das fases de desenvolvimento e ao final do projeto foi possível perceber o quão foi benéfico as estratégias sugeridas, conforme listadas abaixo:

- Para as funcionalidades Pesquisa Avançada, Pesquisa Geral e Gerenciar Administração, foi sugerido pela ferramenta a aplicação de auditoria de código. A auditoria foi realizada para todo o código e foram identificados 163 Bugs e 18% de linhas de código duplicados.

- Para as funcionalidades, Pesquisa Avançada e Geral, Gerenciar Administração e Exportar Resultado foi indicado pela ferramenta a realização de teste de segurança. Na análise foram identificados mais de 1305 vulnerabilidades de segurança, críticas. Com base nesta análise, foi possível o reconhecimento de um erro específico grave, que permite a quebra do *captcha*. Esse elemento foi criado, como uma regra de

segurança, para que não seja possível acesso via robô por hacks indevidamente. Os gestores de Tecnologia da Informação (TI), sinalizaram um valor agregado muito grande, nesta análise.

- Para as funcionalidades de Pesquisa Avançada, Pesquisa Geral, Exportar Resultado da Busca e Visualizar Portal, foi sugerido a realização de testes de performance. Ao realizar testes nas funcionalidades de pesquisa foi setado 100 usuários simultâneos realizando a mesma ação. Neste cenário o percentual de sucesso das requisições foi de 34,99% contra 65,01% de erros. Ao tentar simular 500 usuários simultâneos o percentual de sucesso das requisições foi de 2,27% contra 97,73% de erros. Na funcionalidade de Exportar Resultado da Busca, foi apresentado 10 segundos para *download*, tempo este aceitável para a quantidade de dados baixados. Na funcionalidade de Visualizar Portal, não foi identificado defeitos.

- Para as funcionalidades de Pesquisa Geral e Avançada, foram sugeridas testes de compatibilidade. Neste contexto, foi identificado que para dispositivos móveis a aplicação não se comporta de maneira adequada, apresentado quebra de página.

- Foi surgido também testes de Interface/Usabilidade, em todas as funcionalidades. Como resultado, foi identificado apenas 1 defeito, visto que o cliente não tem um padrão de interface, estabelecido. Neste contexto, foi encomendado pelo cliente a criação de um documento que padronize a interface dos sistemas desenvolvido pelo órgão.

- Para a funcionalidade Visualizar Portal, foi sugerido teste de acessibilidade. Foram identificados 42 erros nessa funcionalidade representando uma porcentagem de 93,37% de uma meta de pelo menos 95%.

- Para todas as funcionalidades, foi sugerido o teste funcional ao qual foram identificados10 incidentes entre a Sprint 1 e 2.

- Os testes de Falha e Recuperação, foram sugerido pelas suas realizações para as funcionalidades Gerenciar Administração e Pesquisa Avançada e Geral, entretanto não foram identificados incidentes para este.

- Foi detectado também a necessidade de realização de Teste Funcional Automatizado para as funcionalidades de Pesquisa Avançada e Pesquisa Geral. Essa ação não foi implantada dentro do projeto, pois era necessário a estabilização do sistema para realizar a automatização.

- Foi sugerido teste de carga, mais não foi realizado, uma vez que o sistema é composto por funcionalidades basicamente de consulta, não tendo inserção de dados, não sendo necessário este tipo de teste.
- O teste integrado ocorreu para o módulo completo do sistema e não foram identificados problemas, após a correção dos incidentes listados.

Desta forma, esta fase mostrou-se bastante efetiva com as indicações dos tipos de testes a serem aplicados no produto de software.

Para o projeto PUMA, ficou evidente que caso a ferramenta fosse aplicada antes, os erros identificados na fase de pré-produção, seriam tratados antecipadamente, sem os impactos causados. Os erros identificados serão tratados pela equipe do PUMA com previsão de liberação para segunda semana de agosto de 2019.

Para o PROJETO 2, a ferramenta proposta se mostrou ainda mais efetiva, pois foi aplicada nas fases de construção permitindo a realização dos testes e correções antes da disponibilização do sistema em produção. O sistema foi disponibilizado em produção no ínicio de julho e após a sua implantação foi identificado apenas 1 incidente que não foi identificado por meio dos testes realizados. Trata-se de um erro de paginação na funcionalidade de Pesquisa Geral. Ao analisar, o motivo do do defeito não ter sido identificado por meio dos testes realizados, foi identificado que no ambiente de teste o erro não ocorre. Isso dá-se ao fato de que o ambiente de teste tem seus dados reduzidos, não permitindo a simulação do erro ocorrido. O erro foi identificado pela equipe interna e não causou impacto a imagem ou ao negócio.

A fase de Monitoramento e Análise Crítica, permitiu um acompanhamento dos incidentes identificados, por meio dos painéis apresentados na ferramenta.

Para o projeto PUMA, é possível identificar 9 incidentes, que encontram-se no seguinte status:

- 3 incidentes resolvidos, voltados a problema de performance. Este foi resolvido, uma vez que foi cedido pela GIGACandanga, projeto que fornece conexões de alta capacidade para as instituições de pesquisa e ensino superior, integradas pela RNP, Rede Nacional de Ensino e Pesquisa, a infraestrutura necessária para armazenamento do sistema.
- 5 incidentes novos e 1 em correção;

Não foi possível a aplicação de todas as correções até a conclusão deste trabalho, pois o financiador do projeto A3M sinalizou a conclusão do valor empenhado para pagamento das bolsas completas, não permitindo assim a atuação da equipe de desenvolvimento nas

correções. Estão sendo verificadas outras possibilidades para a conclusão completa do módulo;

Para o PROJETO 2, todos os incidentes identificados foram corrigidos antes da sua efetiva liberação para produção e esta fase permitiu o monitoramento em tempo de projeto das efetivas correções pela fábrica de software, nos itens identificados.

A última fase de Registro e Reports, promoveu por meio da ferramenta toda a documentação do processo de Gestão de Risco para Qualidade de software, podendo ser acessível e atualizado a qualquer momento.

Destaca-se ainda neste contexto, que a aplicação da ferramenta nas fases de desenvolvimento foi mais efetiva, conforme resultado apresentado ao PROJETO 2, demonstrando assim sua maior efetividade.

Para os resultados é relevante apresentar também, além das fases de aplicação da ferramenta que foram descritas acima os requisitos incorporados a ferramenta, que foram provenientes da pesquisa não estruturada apresentada na seção 4.2.2 e da revisão da literatura demonstrada na seção 4.1 das abordagens já existentes. Considerando o exposto, são apresentados os resultados da implementação, com base na pesquisa não estruturada, foram incorporados à ferramenta os seguintes requisitos:

- Ferramenta de fácil entendimento, para uma participação efetiva da área de negócio;
- Ferramenta web;
- Criação de uma área para gestão dos riscos do produto;
- Ferramenta automatizada, com armazenamento de informações;
- Criação de perguntas como representação das características de qualidade, com vistas a facilitar o entendimento da área técnica e de negócio;
- Apresentação da estratégia de teste;
- Apresentação dos indicadores de teste;
- Ferramenta a ser aplicada no processo de desenvolvimento de software.

Da avaliação da ferramenta de Gestão de Risco para Qualidade de software, com base nos critérios elencados na seção 4.1, tem-se os seguintes resultados:

- No critério simplicidade: para os estudos de caso apresentados, a ferramenta se mostrou de simples aplicação, dentro do processo de desenvolvimento;
- No critério abrangência: foi demonstrado que a construção da ferramenta, foi baseada nos conceitos de qualidade da ISO25010[2], por meio das abas de Identificação de Risco e Análise de Risco;

- No critério efetividade: atende aos critérios, uma vez que ferramenta apresenta por meio do tratamento de riscos as estratégias de teste por funcionalidade;
- No critério redução de esforço: a ferramenta atende e tem um bom atendimento aos critérios, podendo haver alguma restrição, porém de baixo impacto. A matriz de risco, apoia na redução de esforço, por meio da priorização dos testes, entretanto a quantidade de tipos de testes pode elevar o esforço.
- No critério negócio: a ferramenta atende aos critérios de forma adequada, pois a área de negócio é envolvida desde o início no processo e participa efetivamente das avaliações. Destaca-se também que a realização nas fases de desenvolvimento permitiu um conhecimento mais apurado do negócio;
- Processo de gestão de risco: a ferramenta atende aos critérios de forma adequada, dado que teve seu processo de aplicação, pautado na ISO31000[1];
- Tipos de teste: a ferramenta atende aos critérios de forma adequada, considerando que a aba de tratamento de risco, sugere os tipos de testes a serem realizados como estratégia de risco, para reduzir os impactos da má qualidade do software e os estudos de caso apresentados se mostraram bem efetivos a este critério.

Realizando a mesma comparação apresentada na Figura 4.2 aos mesmos critérios considerados na 4.1 a ferramenta de Gestão de Risco para Qualidade de software apresentou uma maior abrangência em detrimento das abordagens e ferramentas já existente, conforme apresentado na Figura 6.1.

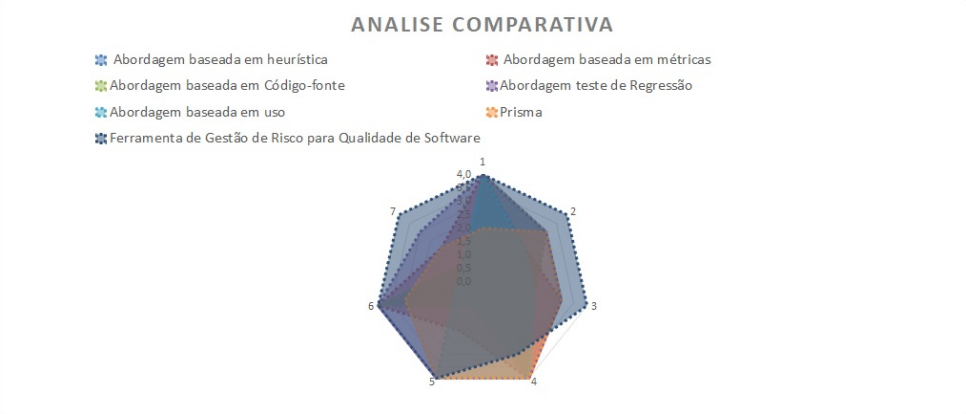



Figura 6.1: Análise Comparativa com a inclusão da ferramenta da pesquisa
Fonte: Elaborado pela autora

Em uma comparação com a Figura x, é possível notar que a cobertura, junto aos critérios definidos, são maiores, uma vez que para todos os critérios elencados, para ferramenta ter uma ampla cobertura, com exceção do critério 4, que se mostrou menos efetiva. Este

critério trata da diminuição do esforço de teste, que para a ferramenta apresentada, não se mostra tão completa, uma vez que a quantidade de testes apresentados para realização são maiores dos que todas as abordagens e ferramenta demostradas na pesquisa. Entretanto, desataca-se que os tipos de testes, apoiaram no aumento da qualidade dos produtos de softwares entregues.

Assim sendo, conclui-se todos os resultados apresentando na próxima seção a futura implantação da ferramenta.

## 6.2 Implantação na RSI Informática

Após a aplicação da ferramenta de Gestão de Risco para Qualidade de software, descrita em detalhes no Capítulo 5, a ferramenta foi apresentada para a Diretoria de Governo e sócio da empresa.

Os resultados causaram boa impressão e a aceitação geral foi imediata. Já foi sinalizado a implantação da ferramenta na organização, que deverá ser estruturada com vistas a ser ofertado por meio de um serviço a sua carteira de clientes.

Foi solicitado a realização de uma apresentação a todos os gerentes de operações da empresa e a montagem de um material mais técnico para apresentação aos clientes, que terá como base os conceitos desta pesquisa. Além disso o cliente do PROJETO 2 apresentou interesse em incorporar a ferramenta dentro da metodologia de desenvolvimento de software, junto a disciplina de teste.

Para a sua correta implantação, será necessária a compra da licença Oracle APEX, pois a pesquisa utilizou a versão de treinamento e ainda disponibilidade de um servidor. Neste sentido, toda infraestrutura será provida pela empresa.

Diante do exposto, fica evidente a aplicabilidade da ferramenta de Gestão de Risco para Qualidade de software no contexto em estudo, uma vez que conseguiu de forma simplificada, por meio da gestão de risco, focar os esforços nas funcionalidades mais críticas e ainda uma estratégia de risco, que permitiu a entrega de um produto de mais qualidade.

No entanto, em uma avaliação mais profunda, é importante aplicar a ferramenta em outros contextos e processos diferentes. Devido a limitações temporais esse objetivo não fez parte do escopo dessa pesquisa, mas poderá ser alvo de estudos futuros.

# Capítulo 7

# Conclusão

## 7.1 Considerações Finais

Diante da importância de entregar softwares com mais qualidade e ter a fase de teste como instrumento de gestão de risco para reduzir as incertezas de um produto com baixa qualidade, este trabalho teve como objetivo desenvolver uma ferramenta de Gestão de Risco para Qualidade de Software, com intuito de gerir os riscos do produto nas fases de desenvolvimento, apoiando na priorização dos testes a serem conduzidos e ainda na estratégia de teste a ser realizada, para assim reduzir a probabilidade e impacto de falhas no produto, permitindo uma entrega de qualidade do produto de software.

Neste contexto, ao finalizar a avaliação dos resultados da ferramenta pode se afirmar que esta pesquisa alcançou seu objetivo de desenvolver uma ferramenta para Gestão de Risco para Qualidade de Software, aplicada as fases de desenvolvimento de software, e validá-la por meio de sua aplicação em 2 estudos de caso distintos, na fases de desenvolvimento por meio de um processo estruturado, objetivando a redução de falhas no produto de software.

A ferramenta proporcionou a gestão de riscos técnicos do produto de software, nas fases de desenvolvimento, tendo como foco a garantia da qualidade do produto, por meio das características de qualidade avaliadas nas funcionalidades do produto. Facilitou, por meio da matriz de riscos apresentadas em quadrantes, focar os esforços de teste nas funcionalidades mais críticas. E ainda uma estratégia de teste de forma automatizada, permitindo uma entrega de produto de software com qualidade.

A adoção da ferramenta demonstrou seu potencial por meio dos benefícios listados abaixo:

- Identificação dos pontos fortes e fracos das abordagens e ferramentas atuais voltadas a gestão de risco para qualidade de software;

- Documentação do processo de gestão de risco para Qualidade Software;
- Identificação dos riscos de produto de Software de forma estrutura;
- Priorização dos esforços de testes;
- Incorporação ao processo de desenvolvimento de Software;
- Avaliação de 2 perspectivas diferentes (Negócio e Técnico) de um único produto;
- Estratégia de teste estruturada, que não depende somente de um gerente de teste;
- Incorporação do processo de gestão de risco;
- Identificação de melhorias no processo de desenvolvimento de software;
- Melhor qualidade dos produtos desenvolvidos, por meio da identificação de defeitos antes da entrega do software;
- Monitoramento e Análise crítica dos defeitos.

Proporcionou ao final da pesquisa, para a RSI Informática, uma ferramenta a ser fornecida aos seus clientes, com intuito de prover um melhor serviço, considerando que seu segmento principal é a garantia da qualidade dos produtos de software. A aplicação da ferramenta, demonstrou por meio dos estudos de caso, a identificação de uma série de defeitos, que pelo processo estabelecido internamente no projeto, não seriam identificados. Permitiu, além da identificação de defeitos no produto, antes do seu envio para a produção, oportunidades de melhorias em seu processo de desenvolvimento, incorporando inclusive a ferramenta proposta nesta pesquisa. Ofereceu ainda, uma gestão de riscos técnicos, ainda em tempo de desenvolvimento, que podem afetar a qualidade do produto, além de permitir uma atuação focada nas funcionalidades mais críticas do projeto. Considerando as pontos expostos, segue listagem de proposta de evolução da ferramenta.

## 7.2 Propostas de estudos futuros

Diante da limitações temporais relativas ao prazo para conclusão desta pesquisa, algumas questões relevantes não fizerem parte do seu escopo, mas poderão ser alvo de estudos futuros. Entre estas questões estão:

- Implantação de técnicas de mineração de dados, para predição de defeitos, por meio de dados, tais como código fonte e defeitos identificados;
- Geração de base de conhecimento de gestão de risco, por segmento de negócio;
- Melhoria na interface da ferramenta;

- Aplicação em projetos de grandes porte;
- Internalização de plano de ação mais estruturado;
- Melhorias nos indicadores por meio de uma única ferramenta;
- Integração direta com as ferramentas de gestão de teste;
- Aplicação da ferramenta em aplicativos móveis.

# Referências


[1] 31000, ISO/IEC: *Iec 31000*. International Organization for Standarization, 2018. vi, 2, 38, 39, 40, 51, 76, 121, 168

[2] 25010, ISO/IEC: *Iec 25010*. International Organization for Standarization, 2006. vi, vii, 2, 24, 25, 26, 27, 28, 29, 30, 75, 82, 99, 103, 107, 124, 125, 161, 167

[3] Lovelock, John David: *Gartner says global it spending to grow 1.1 percent in 2019*. Gartner, STAMFORD, 2019. 1

[4] Mecenas, Ivan e Vivianne OLIVEIRA: *Qualidade de software-uma metodologia para homologação de sistemas*, 2005. 1

[5] Delamaro, Marcio, Mario Jino e Jose Maldonado: *Introdução ao teste de software*. Elsevier Brasil, 2017. 1, 22

[6] Yang, Haijun: *Measuring software product quality with iso standards base on fuzzy logic technique*. Em *Affective Computing and Intelligent Interaction*, páginas 59–67. Springer, 2012. 2

[7] Kaner, Cem, Jack Falk e Hung Quoc Nguyen: *Testing Computer Software Second Edition*. Dreamtech Press, 2000. 2, 3

[8] Pearn, WL e PC Lin: *Computer program for calculating the p-value in testing process capability index cpmk*. Quality and Reliability Engineering International, 18(4):333–342, 2002. 2

[9] MACHADO, Ivan do Carmo: *Riple-te: A software product lines testing process*. 2010. 2

[10] Hall, Elaine M: *Managing risk: Methods for software systems development*. Pearson Education, 1998. 2, 47

[11] Macedo, Mateus Henrique Basso e Eduardo Gomes Salgado: *Gerenciamento de risco aplicado ao desenvolvimento de software*. Sistemas & Gestão, 10(1):158–170, 2015. 2

[12] Aguiar, Mauricio: *Gerenciando riscos nos projetos de software*. Developer's Magazine. Disponível em, 2011. 2

[13] Gusmão, CMG: *Um modelo de processo de gestão de riscos para ambientes de múltiplos projetos de desenvolvimento de software*. Centro de Informática, Universidade Federal de Pernambuco, Recife, 2007. 2, 3, 38

[14] *Solução de problemas de testes baseados em risco.* Software Testing and Quality Engineering, 5(3). 2, 3, 48, 75

[15] Frischknecht, Chelsea, apr 2018. 3, 5

[16] Myers, Glenford J, Corey Sandler e Tom Badgett: *The art of software testing.* John Wiley & Sons, 2011. 3, 30

[17] Moraes, Matheus Henrique Bartolomeu Marques e Francisco Rodrigues Lima Junior: *Proposição e aplicação de uma metodologia baseada no ahp e na iso/iec 25000 para apoiar a avaliação da qualidade de softwares de gestão de projetos.* Revista GEPROS, 12(2):239, 2017. 3, 107

[18] *Early and effective: As vantagens do teste baseado em risco.* Software Test & Performance, 3(7). 3, 6

[19] Rios, Emerson: *Análise de riscos em projetos de teste de software.* Editora Altabooks, 2005. 3

[20] Bach, James: *The challenge of" good enough" software.* 1996. 3, 47, 75, 76

[21] Bach, James: *Heuristic risk-based testing.* Software Testing and Quality Engineering Magazine, 11(9), 1999. 3, 30, 48, 49, 51, 52, 53, 54, 75, 76, 82

[22] Redmill, Felix: *Exploring risk-based testing and its implications.* Software Testing, Verification and Reliability, 14(1):3–15, 2004. 3, 6, 48, 75

[23] Amland, Ståle e Hulda Garborgsv: *Risk based testing and metrics.* Em *5th International Conference EuroSTAR*, volume 99, páginas 8–12, 1999. 3, 54, 55, 77

[24] Rosenberg, Linda H, Ruth Stapko e Albert Gallo: *Risk-based object oriented testing.* 24th SWE, 1999. 3, 58, 78

[25] McMahon, Keith: *Risk-based testing.* ST Labs, WA, 1998. 3, 56

[26] Chen, Yanping, Robert L Probert e D Paul Sims: *Specification-based regression test selection with risk analysis.* Em *Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research*, página 1. IBM Press, 2002. 3, 58, 79

[27] Besson, Stephane: *A strategy for risk-based testing.* na sajtu http://www. sqe. com, 2005. 3, 59, 79

[28] Van Veenendaal, Erik: *Product risk management: the prisma method.* 2009. 3, 60, 61, 63, 80

[29] Garfinkel, Simson: *Piores erros de software da história*, aug 2005. `https://www.wired.com/2005/11/historys-worst-software-bugs/`. 4

[30] Metropole: *Nova falha no sistema zera saldo de correntistas do brb*, may 2018. `https://www.metropoles.com/colunas-blogs/grande-angular/nova-falha-no-sistema-zera-saldo-de-correntistas-do-brb`. 5

[31] Notícias, Diàrio de: *Falha de check-in provoca caos em aeroportos de vários países. os nacionais também foram afetados*, sep 2017. https://www.dn.pt/mundo/interior/falha-de-check-in-provoca-caos-em-aeroportos-de-varios-paises-8804498.html. 5

[32] Globo: *Fiat chrysler detecta falha ligada a uma morte em picapes ram*, may 2017. https://g1.globo.com/carros/noticia/fiat-chrysler-detecta-falha-ligada-a-uma-morte-em-picapes-ram.ghtml. 5

[33] UOL: *Falha no sistema de ti da british airways causa atrasos e cancelamentos de voos*, may 2017. https://adrenaline.uol.com.br/2017/05/27/49867/falha-no-sis/tema-de-ti-da-british-airways-causa-atrasos-e-cancelamentos-de-voos/. 5

[34] Globo: *Boeing reconhece pela primeira vez defeitos no software do simulador de voo do 737 max*, may 2019. https://g1.globo.com/mundo/noticia/2019/05/18/boeing-reconhece-pela-primeira-vez-defeitos-no/-software-do-simulador-de-voo-do-737-max.ghtml. 5

[35] Molinari, Leonardo: *Testes funcionais de software*. Ed. Visual Books. Florianópolis, 66, 2008. 6, 21, 22

[36] *Engenharia de software: Relatório sobre uma conferência patrocinada pelo comitÊ de ciÊncia da otan, garmisch, alemanha, de 7 a 11 de outubro de 1968*. 9

[37] Pressman, Roger S: *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005. 9, 21, 30, 33

[38] Pressman, Roger S: *Engenharia de software*, volume 6. Makron books São Paulo, 1995. 9, 22

[39] Armour, Phillip G.: *Zeppelins and jet planes: a metaphor for modern software projects*. Commun. ACM, 44(10):13–15, 2001. 9, 75

[40] Ruparelia, Nayan B: *Software development lifecycle models*. ACM SIGSOFT Software Engineering Notes, 35(3):8–13, 2010. 10, 12, 13, 15, 16, 18

[41] Macoratti, José Carlos: *O processo de software*. Postado em, 19, 2012. 10

[42] Munassar, Nabil Mohammed Ali e A Govardhan: *A comparison between five models of software engineering*. IJCSI, 5:95–101, 2010. 11, 12, 13, 14, 15

[43] Boehm, Barry W.: *A spiral model of software development and enhancement*. Computer, 21(5):61–72, 1988. 14, 16, 46

[44] *Evoluindo um novo modelo (sdlc model-2010) para o ciclo de vida de desenvolvimento de software (sdlc)*. Jornal Internacional de Ciência da Computação e Segurança de Rede, 10(1). 15

[45] Alshamrani, Adel e Abdullah Bahattab: *A comparison between three sdlc models waterfall model, spiral model, and incremental/iterative model*. International Journal of Computer Science Issues (IJCSI), 12(1):106, 2015. 15, 49, 50, 76

[46] Jacobson, Ivar: *The unified software development process*. Pearson Education India, 1999. 16

[47] Gilb, Tom: *Evolutionary delivery versus the waterfall model*. ACM SIGSOFT Software Engineering Notes, 10(3):49–61, 1985. 16

[48] Anwar, Ashraf: *A review of rup (rational unified process)*. International Journal of Software Engineering, 5(2):8–24, 2014. 16, 17

[49] Kruchten, Philippe: *The rational unified process: an introduction*. Addison-Wesley Professional, 2004. 17

[50] Booch, Grady: *The unified modeling language user guide*. Pearson Education India, 2005. 17

[51] Hirsch, Michael: *Making rup agile*. Em *OOPSLA 2002 Practitioners Reports*, páginas 1–ff. ACM, 2002. 17

[52] Hanssen, Geir Kjetil, Hans Westerheim e Finn Olav Bjørnson: *Using rational unified process in an sme–a case study*. Em *European Conference on Software Process Improvement*, páginas 142–150. Springer, 2005. 17

[53] Beck, Kent, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries *et al.*: *Manifesto for agile software development*. 2001. 18

[54] Beck, Kent: *Embracing change with extreme programming*. Computer, 32(10):70–77, 1999. 18

[55] Schimiguel, Juliano: *Agile development: Xp e scrum em uma abordagem comparativa*, 2017. `https://www.devmedia.com.br/agile-development-xp-e-scrum-em-uma-abordagem-comparativa/30808`. 19

[56] Trendowicz, Adam e Sylwia Kopczyńska: *Adapting multi-criteria decision analysis for assessing the quality of software products. current approaches and future perspectives*. Em *Advances in COMPUTERS*, volume 93, páginas 153–226. Elsevier, 2014. 20

[57] Chandrupatla, Tirupathi R: *Quality and reliability in engineering*, volume 2. Cambridge University Press New York, NY, 2009. 20

[58] Crosby, Philip B: *Qualidade sem lágrimas: a arte da gerência descomplicada*. Rio de Janeiro: José Olimpio, 1992. 21

[59] Rawashdeh, Adnan e Bassem Matalkah: *A new software quality model for evaluating cots components*. Journal of Computer Science, 2(4):373–381, 2006. 22

[60] Boas, Vilas e André Luis de Castro: *Qualidade e avaliação de produto de software.* Em *Textos Acadêmicos, Curso de Pós-Graduação "Lato Sensu"/(Especialização) a Distância: Melhoria de Processo de Software, Lavras (MG)–UFLA/FAEPE*, página 58, 2004. 22

[61] Al-Qutaish, Rafa E: *Quality models in software engineering literature: an analytical and comparative study.* Journal of American Science, 6(3):166–175, 2010. 23, 52, 75

[62] Behzadian, Majid, Reza Baradaran Kazemzadeh, Amir Albadvi e Mohammad Aghdasi: *Promethee: A comprehensive literature review on methodologies and applications.* European journal of Operational research, 200(1):198–215, 2010. 23

[63] Boehm, Barry W, John R Brown e Hans Kaspar: *Characteristics of software quality.* 1978. 23, 75

[64] Dromey, R. Geoff: *A model for software product quality.* IEEE Transactions on Software Engineering, 21(2):146–162, 1995. 23

[65] 9126-4, ISO/IEC: *ISO/IEC 9126-4. Software engineering – Product quality.* ISO/IEC 9126, 2001. 23, 24, 52, 75

[66] Miguel, José P, David Mauricio e Glen Rodríguez: *A review of software quality models for the evaluation of software products.* arXiv preprint arXiv:1412.2977, 2014. 30

[67] Bastos, Anderson, Emerson Rios, Ricardo Cristalli, Trayahú Moreira *et al.*: *Base de conhecimento em teste de software.* São Paulo, 2007. 30, 32

[68] Graham, Dorothy, Erik Van Veenendaal e Isabel Evans: *Foundations of software testing: ISTQB certification.* Cengage Learning EMEA, 2008. 30, 31, 33, 75

[69] Bernardo, Paulo Cheque e Fabio Kon: *A importância dos testes automatizados.* Engenharia de Software Magazine, 1(3):54–57, 2008. 34

[70] Muller, Tomas e D Friedenberg: *Certified tester foundation level syllabus.* Journal of International Software Testing Qualifications Board, 2011. 36

[71] 31010, ISO/IEC: *Iec 3101 - 0risk management – risk assessment techniques.* International Organization for Standarization, 2009. 43

[72] Gusmão, Cristine Martins Gomes de: *Um modelo de processo de gestão de riscos para ambientes de múltiplos projetos de desenvolvimento de software.* 2009. 44

[73] Abreu Pinna, Cristina Coelho de e Marly Monteiro de Carvalho: *Gestão de escopo em projetos de aplicações web.* Revista Produção Online, 8(1), 2008. 44

[74] Paulk, Mark: *Capability maturity model for software.* Encyclopedia of Software Engineering, 2001. 44

[75] Souza, E e C Gusmão: *Rbtprocess: Modelo de processo de teste de software baseado em riscos.* Em *13º WTES-Workshop de Teses e Dissertações em Engenharia de Software*, 2008. 45, 76, 77, 78, 79

[76] Boehm, Barry W: *Software risk management: principles and practices.* IEEE software, 8(1):32–41, 1991. 45, 46

[77] Fairley, Richard: *Risk management for software projects.* IEEE software, 11(3):57–67, 1994. 45

[78] Team, CMMI Product: *Capability maturity model® integration (cmmi), version 1.1–continuous representation.* 2002. 45

[79] Kruchten, Philippe: *Introdução ao RUP: rational unified process.* Ciência Moderna, 2003. 45

[80] PMBOK, GUIDE: *Um guia do conhecimento em gerenciamento de projetos.* Sexta Edição, 6:407, 2017. 46, 93

[81] Pfleeger, SL e JD Palmer: *Software estimation for object-oriented systems.* Em *1990 Int. Function Point Users Group Fall Conf*, páginas 181–196, 1990. 57

[82] Chidamber, Shyam R e Chris F Kemerer: *Towards a metrics suite for object oriented design*, volume 26. ACM, 1991. 57

[83] McCabe, Thomas J: *A complexity measure.* IEEE Transactions on software Engineering, (4):308–320, 1976. 57

[84] Campbell, G e Patroklos P Papapetrou: *SonarQube in action.* Manning Publications Co., 2013. 58

[85] Chen, Yanping: *Specification-based regression testing measurement with risk analysis.* Tese de Doutoramento, University of Ottawa (Canada), 2003. 59, 79

[86] Van Veenendaal, Erik: *The PRISMA approach.* Uitgeverij Tutein Nolthenius, 2012. 62

[87] Ciribelli, Marilda Corrêa: *Como elaborar uma dissertação de mestrado através da pesquisa científica.* 7Letras, 2003. 66

[88] Gerhardt, Tatiana Engel e Denise Tolfo Silveira: *Métodos de pesquisa.* Plageder, 2009. 66

[89] Gil, Antonio Carlos: *Como elaborar projetos de pesquisa.* São Paulo, 5(61):16–17, 2002. 66

[90] Flores, Maria Lucia Pozzatti: *O uso do excel para resolver problemas de operações financeiras.* RENOTE, 2(2), 2004. 72

[91] Office: *Microsoft excel*, may 2019. `https://products.office.com/pt-br/excel`. 72

[92] Monger, Alastair, Sheila Baron e Jing Lu: *More on oracle apex for teaching and learning.* 2009. 72

[93] Bittencourt, Marcelo: *Gestão de teste e defeitos com testlink e mantis.* 1º Edição, 1, 2013. 72

[94] Fuentes-Fernández, Lidia e Antonio Vallecillo-Moreno: *An introduction to uml profiles.* UML and Model Engineering, 2:6–13, 2004. 72

[95] PowerDesigner:, SAP: *Ferramentas de arquitetura empresarial para o sucesso da transformação digital*, may 2019. `https://www.sap.com/brazil/products/powerdesigner-data-modeling-tools.html`. 73

[96] BI:, Power: *Business intelligence como nunca*, 2019. `https://powerbi.microsoft.com/pt-br/`. 73

[97] ATLASSIAN: *O trello permite trabalhar com mais colaboração e ter mais produtividade*, 2019. `https://trello.com`. 73

[98] Likert, Rensis: *A technique for the measurement of attitudes.* Archives of psychology, 1932. 76

[99] Welter Neto, José Arnildo: *Teste de software em um ambiente virtual de aprendizagem moodle: uma abordagem utilizando risk based testing.* 2011. 76

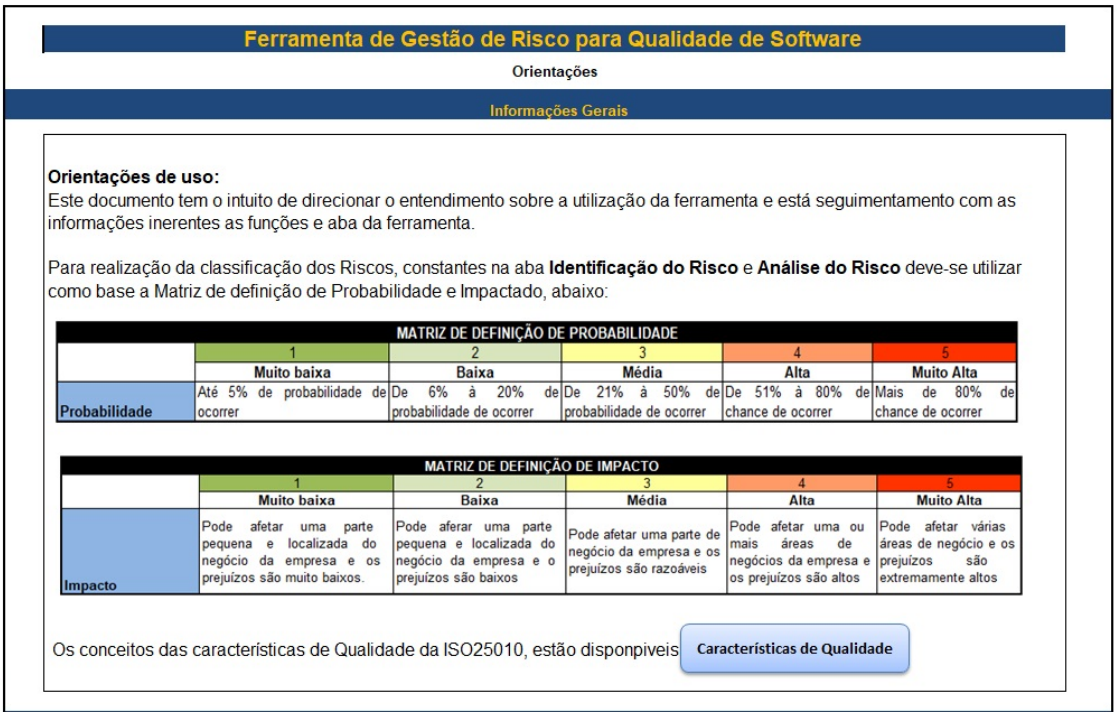
[100] Jørgensen, Lars Kristoffer Ulstein: *A software tool for risk-based testing.* Trabalho de Graduação, Norwegian University of Science and Technology, Norway, 2004. 77, 79

[101] Oracke: *Application express developer's guide*, 2009. `https://docs.oracle.com/cd/E24693_01/appdev.11203/e11946/intro_app.htm`. 115

[102] Monteiro, Simone Borges Simão, Marcito Ribeiro Madeira Campos, Ana Cristina Fernandes Lima e Ari Melo: *Evaluating direct and indirect results of the active methodology in learning: proposal of an integrative design in 360º via unified platform.* 133

# Apêndice A

# Manual de Instrução da Ferramenta



**Ferramenta de Gestão de Risco para Qualidade de Software**

Orientações

Informações Gerais

**Orientações de uso:**
Este documento tem o intuito de direcionar o entendimento sobre a utilização da ferramenta e está seguimentamento com as informações inerentes as funções e aba da ferramenta.

Para realização da classificação dos Riscos, constantes na aba **Identificação do Risco** e **Análise do Risco** deve-se utilizar como base a Matriz de definição de Probabilidade e Impactado, abaixo:

MATRIZ DE DEFINIÇÃO DE PROBABILIDADE

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | **Muito baixa** | **Baixa** | **Média** | **Alta** | **Muito Alta** |
| **Probabilidade** | Até 5% de probabilidade de ocorrer | De 6% à 20% de probabilidade de ocorrer | De 21% à 50% de probabilidade de ocorrer | De 51% à 80% de chance de ocorrer | Mais de 80% de chance de ocorrer |

MATRIZ DE DEFINIÇÃO DE IMPACTO

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | **Muito baixa** | **Baixa** | **Média** | **Alta** | **Muito Alta** |
| **Impacto** | Pode afetar uma parte pequena e localizada do negócio da empresa e os prejuízos são muito baixos. | Pode aferar uma parte pequena e localizada do negócio da empresa e o prejuízos são baixos | Pode afetar uma parte de negócio da empresa e os prejuízos são razoáveis | Pode afetar uma ou mais áreas de negócios da empresa e os prejuízos são altos | Pode afetar várias áreas de negócio e os prejuízos são extremamente altos |

Os conceitos das características de Qualidade da ISO25010, estão disponpiveis Características de Qualidade

7

Figura A.1: Manual da ferramenta - Parte I
Fonte: Elaboradora pela autora

**Preenchimento da ferramenta**

| Função | Cadastro |
|---|---|
| Deve ser cadastrado as seguintes informações: Função, Partes Interessadas, Associação das Partes Interessadas as Funções relacionadas aos projetos, Clientes, Segmento de Negócio, Tipo de Público Alvo e Tipo do Risco. Todos os cadastros realizados nessa funcionalidade, são refletidos na função de projetos. | |

| Função | Projeto |
|---|---|
| Deve ser cadastrado o projeto a ser avaliado pela ferramenta. Deve ser informado a sigla e nome do projeto, Responsável, Segmento de Negócio, Cliente e Público alvo. As informações listadas nesta tela são provenientes da função cadastro.<br><br>Necessário lenvantar junto ao gerente e patronicador do projeto de software as informações gerais inerente ao produto de software em construção. Deve ser identificado todas as partes interessadas que irão participar do processo de Gestão de Risco com o intuito de garantir a Qualidade de Software, sendo necessário a participação com maior conhecimento no produto de software, do ponto de vista de desenolvimento e negócio. | |

| Aba | Partes Interessadas |
|---|---|
| Deve ser listada todas as partes interessadas que irão participar do processo de Gestão de Risco para Qualidade de Software. | |

| Aba | Funcionalidades do Projeto |
|---|---|
| Deve ser listado todas as funcionalidades do produto em avaliação, que são identificadas pela ferramenta como itens de riscos. | |

| Aba | Identificação do Risco |
|---|---|
| Para garantir a qualidade do produto de software a ferramenta de Gestão de Risco para Qualidade de Software, está baseada na ISO25000, e aborda fatores de qualidade (Interna e Externa) e qualidade em uso de produto.<br><br>Nesta aba, deve-se ser identificado todos os eventos de riscos referente ao produto de software, através de técnicas de levantamento de riscos (Entrevistas, Reunioes e Brainstorming). A identificação dos riscos e a classificação dos riscos impactarão diretamente a aba de **Análise do Risco**, ao qual serão atribuidos pesos às perguntas referentes as caracteristicas de qualidade de um produto de software. | |

| Aba | Análise do Risco |
|---|---|
| Trata-se de uma avaliação dos riscos, baseado na complexidade e na criticidade do negócio das funcionalidades do software, nas áreas propensas a mais defeito. Desta forma:<br><br>1. Deve ser listado todas as funcionalidades do produto de software que são os fatores de riscos. Este formulário deve ser aplicado a todos os Itens de riscos (funções/requisitos) que compõem o projeto de software em avaliação.<br>2. **Probabilidade e impacto** são variáveis independentes, que serão identificadas e avaliadas levando em consideração as características de qualidade, através das perguntas que são mapeadas na ferramenta, sendo que:<br><br>**Probabilidade** - Riscos Técnicos - possibilidade ou chance de um evento de risco ocorrer em um produto de software que deve ser identificado pela equipe de desenvolvimento. Deve ser considerado a (matriz de definição de probabilidade) para a pontuação dos fatores de riscos Vs perguntas.<br>**Impacto** - Riscos de Negócio - é o efeito no produto de software se o evento ocorrer e deve ser identificado pelos usuários de negócio. Deve ser considerado a (matriz de definição de probabilidade) para a pontuação dos fatores de riscos Vs perguntas.<br><br>Os pesos atribuídos às perguntas da aba Análise do Risco, são calculados a partir da soma da pontuação da Probabilidade e Impacto, sendo que quando não for atribuído ao risco um critério de qualidade a pergunta terá peso 1.<br>Após a definição da probabilidade e impacto, acionar a salvar. | |

| Aba | Matriz de Priorização |
|---|---|
| Com base na Análise do Risco realizada, será gerada uma matriz de priorização, confome quadrantes abaixo:<br><br>**Quadrante**<br>I — Baixo impacto/Baixa probabilidade.<br>II — Baixo Impacto/Alta Probabilidade.<br>III — Alto Impacto/Baixa probabilidade<br>IV — Alto Impacto/Alta Probabilidade | |

| Aba | Tratamento do Risco |
|---|---|
| Trata-se da estratégia de teste de software para mitigar os riscos da má qualidade do software, que é agrupado nas seguintes vertentes:<br>**1. Priorização das funcionalidades (Fatores de Riscos)**<br>Será listado todas as funcionalidades mapeadas na Análise de Riscos, apresentando as funcionalidades elencados pelos mais prioritários aos menos prioritários.<br>**2. Estratégia de testes para mitigação de riscos da má qualidade do Software**<br>É apresentado de forma automática a estratégia para mitigar o risco da má qualidade do software. Esta estratégia é baseada na classificação de probabilidade e impacto respondidas nas perguntas associadas às características de qualidade. Considerando ser o teste a disciplina da engenharia de software que mitiga os riscos da má qualidade do software, a ferramenta lista todos os tipos de teste que podem ser aplicados agrupados por funcionalidade. | |

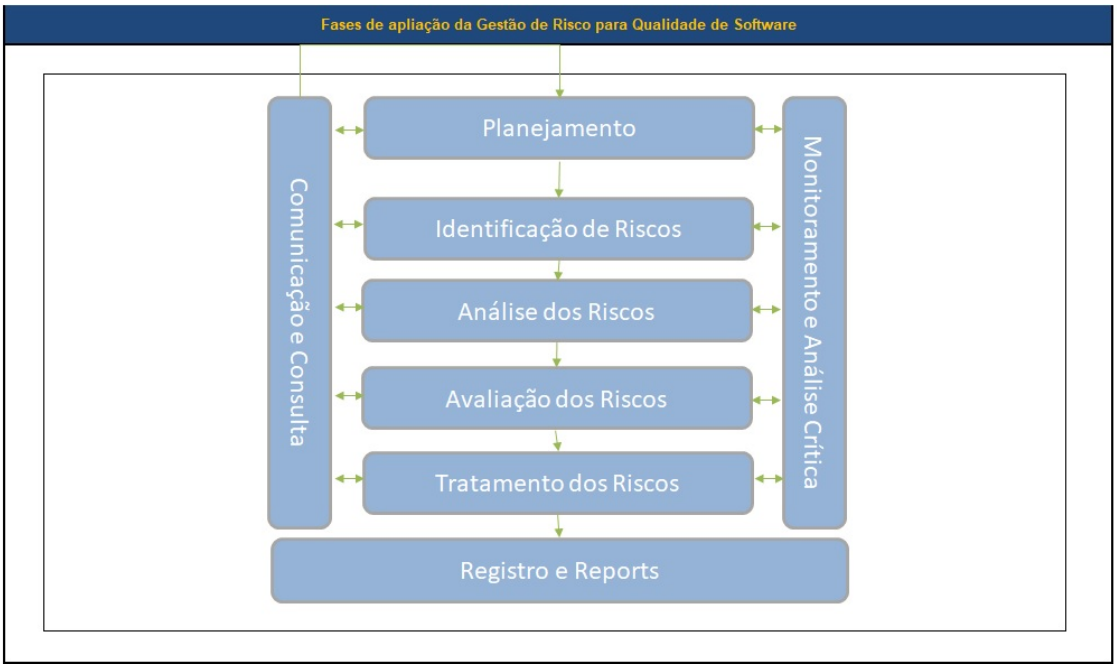



Figura A.3: Manual da ferramenta - Parte III
Fonte: Elaboradora pela autora

# Anexo I

# Incidentes registrados para PUMA

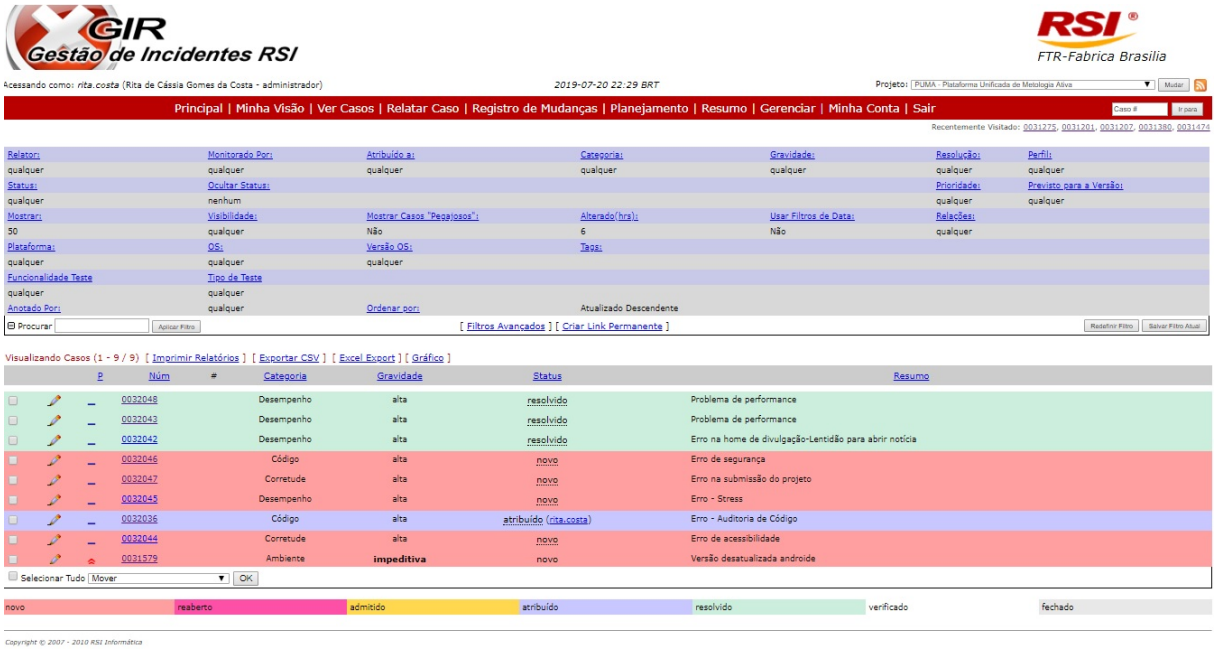



Figura I.1: Incidentes identificados para estudo de caso - PUMA
Fonte: RSI Informática

# Anexo II

# Incidentes registrados para o PROJETO 2

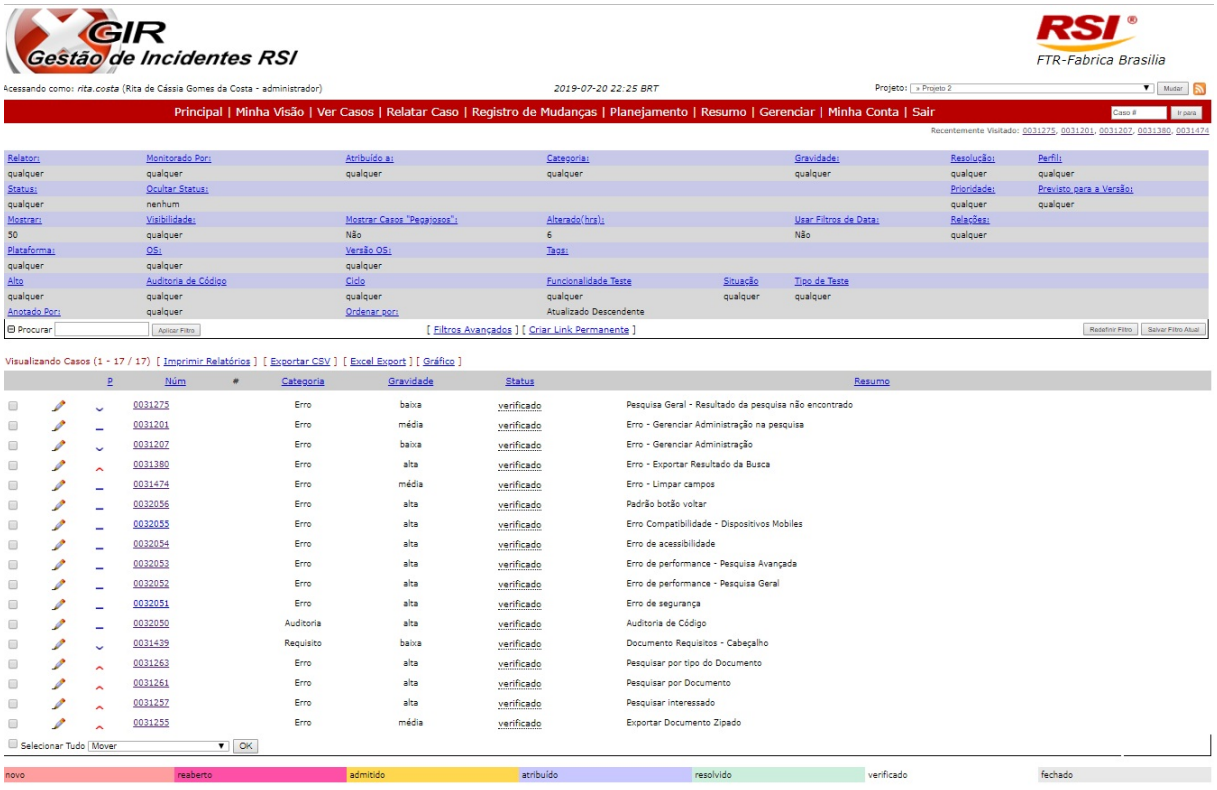



Figura II.1: Incidentes identificados para o estudo de caso - PROJETO 2
Fonte: RSI Informática

# Anexo III

# Status dos Defeitos apresentados nos Painéis BI

# 1. Status dos Painéis BI

Os tópicos descrevem cada um dos estados da figura a seguir, assim como as orientações para alteração dos estados ao longo do fluxo da ferramenta Mantis, que serão apresentados nos Painéis de BI da funcionalidade de Monitoramento e Análise Critica.


## 1.1 Novo

O status “Novo” representa a abertura de um defeito ou não conformidade, encontrado pela equipe de Teste e Qualidade.

Os defeitos com esse estado devem ser avaliados, em seguida ser atribuído a um responsável pela resolução do problema registrado.

## 1.2 Retorno

O status de “Retorno” representa a situação de reincidência do defeito, ou seja, a solução não atende as especificações. Nestas situações, o incidente deve ser atribuído de volta para o responsável pela resolução do problema identificado.

Este estado também pode ser utilizado quando o responsável pelo tratamento identificar mais informações, ou melhor, esclarecimento do incidente registrado.

## 1.3 Em Espera

O status “Em Espera” representa que o defeito registrado foi corrigido pelo responsável atribuído, porém só será liberada para validação da equipe de Teste e Qualidade quando for liberada uma nova versão atualizada da aplicação.

## 1.4 Adiado

O status “Adiado” representa que o defeito registrado foi constatado que realmente é um defeito, mas não será corrigido na versão atual do software. Neste caso deverá ser feita a justificativa por adiar a correção do incidente no campo “Anotação” do caso Mantis.

## 1.5 Em correção

O status “Em correção” representa que o defeito está com o responsável para a correção satisfatória do problema registrado.

## 1.6 Resolvido

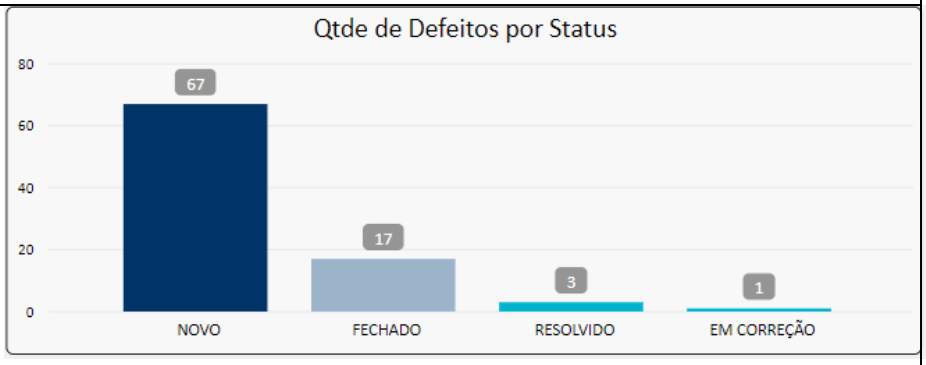
O status “Resolvido” representa que o defeito foi corrigido pelo responsável atribuído. Nestes casos, devem-se realizar os testes de confirmação pela equipe de Teste e Qualidade para verificar se o erro tenha sido realmente corrigido.

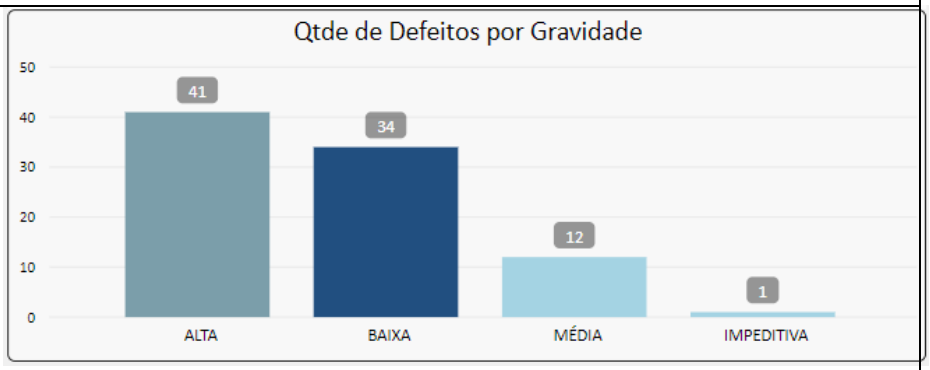
## 1.7 Fechado

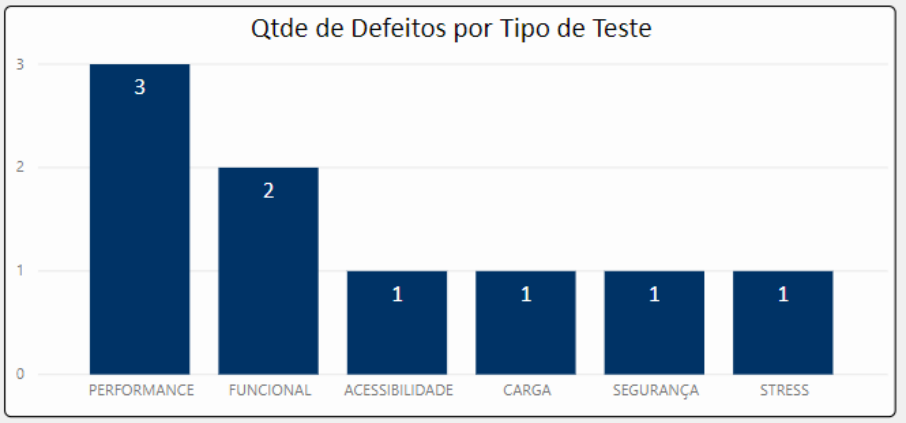
O Status “Fechado” que representa o fechamento de defeito após execução dos testes de confirmação para validar a correção realizada. Caso o erro tenha sido realmente corrigido, a equipe irá colocar a solicitação no estado final FECHADO e seleciona o motivo de resolução adequado (no caso, corrigido). Quando a correção não tenha sido satisfatória, a equipe de Teste e Qualidade Quando deve então mudar o estado da solicitação para RETORNO e atribuir de volta ao responsável pela resolução.

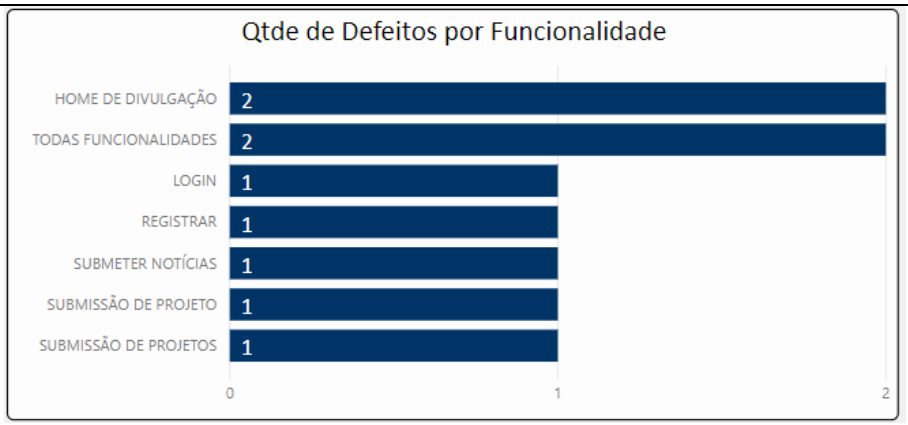
# Anexo IV

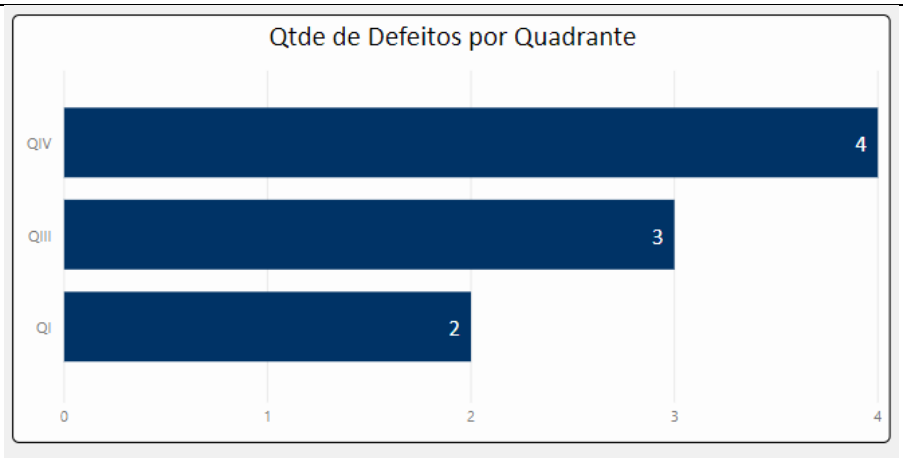
# Indicadores

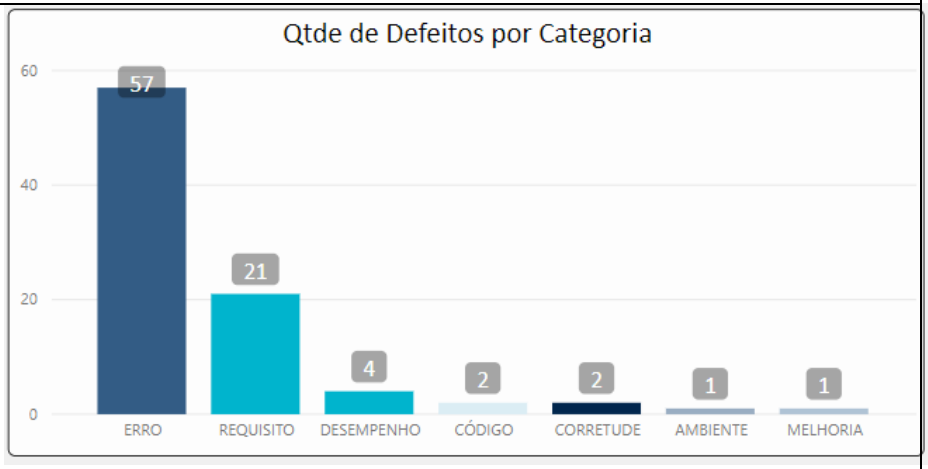
<table>
<tr><th colspan="2">Qte de Defeitos por Status</th></tr>
<tr><td>Objetivo da Medição:</td><td>Acompanhar a evolução da quantidade de defeitos por situação no projeto.</td></tr>
<tr><td>Fonte:</td><td>Ferramenta de Gestão de Incidentes Mantis.</td></tr>
<tr><td rowspan="2">Fórmula:</td><td>$i_m = \sum incidente$</td></tr>
<tr><td>Onde:<ul><li>$i_m$ = quantidade de defeitos registrados no Mantis para o projeto;</li><li>$incidente_{novo}$ = quantidade de defeitos novos, recém criados pelo testador;</li><li>$incidente_{atribuído}$ = quantidade de defeitos em correção;</li><li>$incidente_{resolvido}$ = quantidade de defeitos resolvidos;</li><li>$incidente_{fechado}$ = quantidade de defeitos fechados pelo testador;</li><li>$incidente_{retorno}$ = quantidade de defeitos retornados pelo testador para correção (problema que persiste);</li><li>$incidente_{adiado}$ = quantidade de defeitos adiados e que não serão resolvidos no ciclo de testes corrente;</li><li>$incidente_{em\ espera}$ = quantidade de defeitos resolvidos do ponto de vista do analista/desenvolvedor, mas que ainda não serão liberados para que o testador valide.</li></ul></td></tr>
<tr><td>Meta:</td><td>$i_m \leq 0$</td></tr>
<tr><td>Forma de Apresentação:</td><td> 
</td></tr>
</table>

| **Qte. Defeitos por Gravidade** | |
|---|---|
| Objetivo da Medição: | Acompanhar a evolução da quantidade de defeitos por gravidade |
| Fonte: | Ferramenta de Gestão de Incidentes Mantis. |
| Fórmula: | $i_m = \sum incidente$ |
| | Onde:<br>• $i_m$ = quantidade de defeitos registrados no Mantis para o projeto;<br>• $incidente_{baixa}$ = quantidade de defeitos com gravidade baixa: não inviabiliza a execução do teste;<br>• $incidente_{média}$ = quantidade de defeitos com gravidade média: inviabiliza parcialmente a correta execução do teste, mas o resultado final não é comprometido;<br>• $incidente_{alta}$ = quantidade de defeitos com gravidade alta: inviabiliza a correta execução do teste comprometendo o resultado final;<br>• $incidente_{impedimento}$ = quantidade de defeitos caracterizados como travamento: interrompe a execução dos testes, sendo possível completar o teste por um caminho diferente do ideal; |
| Meta: | $i_m \leq 0$ |
| Forma de Apresentação: |  |

| **Qte. Defeitos por Tipo de Teste** | |
|---|---|
| Objetivo da Medição: | Acompanhar a evolução dos tipos de testes definidos na ferramenta de Gestão de Riscos para Qualidade de Software. |
| Fonte: | Ferramenta de Gestão de Incidentes Mantis. |
| Fórmula: | $i_m = \sum incidente$ |
| | Onde:<br>• $i_m$ = quantidade de defeitos registrados no Mantis para o projeto;<br>• *incidente* = quantidade de defeitos classificados por tipo de teste, englobando: (Carga, Integridade de Dados, Funcional, Acessibilidade, Auditório de Código, Stress, Interface/Usabilidade, Compatibilidade, Performance, Integração, Segurança, Recuperação, Automatizado, Código, Acessibilidade. |
| Meta: | $i_m \leq 0$ |
| Forma de Apresentação: |  |

| **Qte. Defeitos por Funcionalidades** | |
|---|---|
| Objetivo da Medição: | Acompanhar a evolução da quantidade de defeitos por funcionalidade de acordo com Análise de Risco |
| Fonte: | Ferramenta de Gestão de Incidentes Mantis. |
| Fórmula: | $i_m = \sum incidente$ |
| | Onde:<br>• $i_m$ = quantidade de defeitos registrados no Mantis para o projeto, categorizado por funcionalidade. |
| Meta: | $i_m \leq 0$ |
| Forma de Apresentação: |  |

| **Qte. Defeitos por Quadrantes** | |
|---|---|
| Objetivo da Medição: | Acompanhar a evolução da quantidade de defeitos por quadrante de acordo com a Análise de Risco. |
| Fonte: | Ferramenta de Gestão de Incidentes Mantis. |
| Fórmula: | $i_m = \sum incidente$ |
| | Onde:<br>• $i_m$ = quantidade de defeitos registrados no Mantis para o projeto, categorizado por Quadrante<br>• *QI, QII, QIII e QIV* |
| Meta: | $i_m \leq 0$ |
| Forma de Apresentação: |  |

| **Qte. Defeitos por categoria** | |
|---|---|
| Objetivo da Medição: | Acompanhar a evolução da quantidade de defeitos por classificação funcional, que relaciona o erro a uma inconformidade em nível de usabilidade, sistema (aplicação não está de acordo com a documentação), layout visual, documentação ou ambiente no projeto. |
| Fonte: | Ferramenta de Gestão de Incidentes Mantis. |
| Fórmula: | $i_m = \sum incidente$ |
| | Onde:<br>• $i_m$= quantidade de defeitos registrados no Mantis por categoria dos erros identificados. |
| Meta: | $i_m \leq 0$ |
| Forma de Apresentação: |  |