UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Bruno Coelho Lopes

Geração Automática de Relatórios de Confiabilidade: Previsão de falhas de software com base nos padrões de eventos múltiplos na Plataforma X-RAT.

Uberlândia, Brasil

2023

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Bruno Coelho Lopes

Geração Automática de Relatórios de Confiabilidade: Previsão de falhas de software com base nos padrões de eventos múltiplos na Plataforma X-RAT.

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Rivalino Matias Júnior

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2023

Dedico este trabalho aos meus pais, por toda atenção, amor e carinho, sempre presentes em todos os momentos.

Agradecimentos

Ao professor Rivalino Matias Júnior, por toda a motivação e orientação, facilitando a realização deste trabalho.

Ao Caio Santos e Diogo Peixoto, este trabalho não seria possível sem o suporte e paciência de ambos.

Ao meu pai, minha mãe, meus irmãos, minha namorada e meus amigos, pelo apoio e incentivo que serviram de alicerce para as minhas realizações.

Resumo

Durante a execução de softwares encontramos incertezas quanto ao número de faltas ou defeitos remanescentes no software após a sua implementação. Mesmo quando ao analisar este software, ele tenha uma confiabilidade próxima ao limite máximo, não há a certeza de que o produto não apresentará falhas a qualquer momento durante a sua execução.

A confiabilidade é um atributo de extrema importância do software, ao desconsiderá-la pode implicar na possível ocorrência de falhas durante a execução do sistema.

Este trabalho tem como objetivo desenvolver a segunda versão da ferramenta de análise e geração automática de laudos de confiabilidade do Windows 10 e 11 para a plataforma X-RAT (X-Reliability Analysis Tool).

Apresentando um estudo de cenário que mostra os resultados obtidos após complementos e acréscimo de novas análises feitas pela ferramenta, concluindo com a geração de um relatório para o usuário final.

Palavras-chave: Confiabilidade de Software, Sistemas Operacionais, Relatório de confiabilidade.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

.SDLAB	Software Dependability Laboratory
HPDCS	High Performance and Dependable Computing Systems
logs	Registro das falhas de um determinado sistema
MTBF	Mean Time Between Failures
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
OsRat	The Operating System Reliability Analysis Tool
PFC	Padrão de Falha Candidata
SO_{APP}	Aplicação de Sistema Operacional
SO_{KNL}	Kernel
SO_{SVC}	Serviço de Sistema Operacional
SO	Sistema Operacional
TBF	Time Between Failures
$USER_{APP}$	Aplicação de Usuário
Win10	Windows 10
Win11	Windows 11
Win7	Windows 7
X-RAT	X-Reliability Analysis Tool

Sumário

1 Introdução

A tecnologia dominou completamente o cotidiano, principalmente durante a pandemia do COVID-19, onde grande parte da população se viu obrigada a realizar suas obrigações sem sair de casa (trabalhar, estudar, realizar compras e outros afazeres do dia a dia). Com isso, evolução nos softwares foi gigante, e mais do que nunca, estão presentes em todos os lugares, seja em simples aplicações de entretenimento, compras, aplicações de produtividade, passando por softwares mais avançados destinados à área de engenharia, à área hospitalar, inteligência artificial, entre outros de alta precisão, onde os dados gerenciados são críticos e vidas dependem deles.

Um sistema crítico no mundo atual é aquele sistema em que pessoas e/ou organizações dependem, onde uma falha pode acarretar em grandes prejuízos. Os sistemas críticos são altamente dependentes de software de qualidade e confiável, necessitando que tenha uma boa construção, operação e manutenção, gerenciado por profissionais capacitados.

Existem três tipos de sistemas críticos:

- Sistema crítico de segurança: Qualquer falha nesses sistemas resulta em ferimentos, morte ou danos ao meio ambiente. Por exemplo, sistema de planta química (TAWDE, 2022).

- Sistema de missão crítica: Qualquer falha nesses sistemas resulta na falha de alguns objetivos esperados. Por exemplo, sistema de navegação de naves espaciais.

- Sistema crítico de negócios: Qualquer falha nesses sistemas resulta em grandes perdas financeiras. Por exemplo, sistemas de contabilidade utilizados em bancos.

É fato que na sociedade moderna existe a ampla dependência dos sistemas computacionais, falhas nesses sistemas podem gerar desde simples inconvenientes até grandes desastres, seguem alguns exemplos:

- Boeing 737 MAX

De acordo com (RICARDO, 2022), a queda do Boeing 737 MAX, que voava para Nairobi, seis minutos após a decolagem de Addis Abeba, matou todos os passageiros e tripulantes a bordo e exigiu a retenção no solo, por vinte meses, de todos os 737 MAX que estavam em operação, gerando a pior crise da história da Boeing. O acidente ocorreu apenas alguns meses após a queda, em outubro de 2018, de um 737 MAX operado pela empresa indonésia Lion Air. Esse novo acidente matou 189 pessoas e aconteceu momentos depois o avião decolar de Jacarta. Ambos os acidentes foram similares: os aviões mergulharam para o solo descontrolados. Os investigadores atribuíram esses movimentos por falhas em um sistema de software de prevenção de estol, que empurrou o nariz dos aviões

para baixo, após fazerem subidas e descidas erráticas, antes de caírem. Além dos prejuízos decorrentes da paralisação das entregas dos aviões, a Boeing acabou dispendendo cerca de US$ 3 bilhões para pagamento de indenizações e multas e, ao que parece, as disputas judiciais ainda não estão totalmente encerradas.

- CIA distribui gás aos soviéticos

Agentes da CIA supostamente plantaram um *bug* em um sistema de computador canadense comprado pelos soviéticos para controlar seus gasodutos. A compra fazia parte de um plano estratégico soviético para roubar ou obter secretamente tecnologia sensível dos EUA. Quando a CIA descobriu a compra, eles sabotaram o software para que passasse pela inspeção soviética, mas falhasse na operação. O software perdeu o controle e produziu uma intensa pressão no gasoduto Trans-Siberian, resultando na maior explosão não-nuclear da história (RIBEIRO, 2012). Os desastres mencionados acima foram causados por falhas de software, por isso garantir a qualidade do software tornou-se um requisito de extrema importância, evitando que ocorra mal funcionamento ou com inatividades não planejadas. A confiabilidade de software é a probabilidade de operação de software desempenhar com êxito seu propósito especificado por um determinado período (ANSI/IEEE, 1990). Diante deste cenário, a plataforma X-RAT (*X-Reliability Analysis Tool*) (THE. . . , 2020) foi desenvolvida pelo grupo de pesquisas Software Dependability Laboratory (.SDLAB) (SOFTWARE. . . , 2020), com o intuito de automatizar o processo de análise de falhas de software e estimar as métricas de confiabilidade, realizando a coleta, organização e análise dos *logs* de falhas com o objetivo final de produzir um relatório de confiabilidade para o usuário final.

1.1 Justificativa

A confiabilidade de um software é um dos pilares para que se possa ter um bom produto final ao cliente, por isso, a preocupação com uma baixa e/ou nenhuma taxa de falha deve ser grande, tanto nos sistemas críticos, quanto em sistemas de modo geral, pois são eles que estão presentes na maioria das aplicações utilizadas no dia a dia. Por este motivo, ter uma boa confiabilidade durante todo o período de uso da aplicação é tão importante.

Entretanto, existem vários fatores que influenciam na confiabilidade de um software. A plataforma em que a aplicação será executada é um dos principais pontos a serem analisados, visto que, se o sistema operacional em si falhar, a operação terá o mesmo destino. Sendo assim é necessário verificar também a confiabilidade do sistema operacional que suportará a aplicação. Devido aos sistemas operacionais Microsoft Windows 10 e 11 serem os mais utilizado atualmente nos *desktops*, estando presente em 62,65% dos dispositivos *desktop*, este trabalho foi realizado com foco nestes Sistemas Operacionais

(STATCOUNTER..., 2023).

Por serem sistemas operacionais lançados recentemente, possuem poucos trabalhos que mensuram a confiabilidade destes sistemas e seus aplicativos. Com isso, a motivação do trabalho se faz no objetivo de dar continuidade ao trabalho realizado em (PEIXOTO, 2023), que desenvolveu uma importante ferramenta nesta área, que estima automaticamente métricas de confiabilidade de software com base em dados de *logs* e gera um relatório de confiabilidade com base nas métricas de confiabilidade.

1.2 Objetivos

1.2.1 Geral

O objetivo deste projeto é aprimorar com novas análises a ferramenta X-RAT, que implementa as técnicas utilizadas em confiabilidade de software, que são aplicadas para analisar dados de falha dos sistemas operacionais Windows 10 (Win10) e Windows 11 (Win11).

1.2.2 Específicos

Baseando-se no objetivo geral, podemos definir os seguintes objetivos específicos:

- Acrescentar novas análises nos dados dos eventos de falha;
- Realizar a previsão de falha de software com base em padrões de eventos múltiplos;
- Aprimorar o design do relatório final para melhor compreensão do usuário.

1.3 Metodologia

Este trabalho faz parte da continuação do desenvolvimento da aplicação X-RAT (THE..., 2020) do .SDLAB(SOFTWARE..., 2020), conforme foi descrito na Subseção 1.2.1. Continuará sendo utilizada toda a base de construção do software realizada na versão 1 do relatório (PEIXOTO, 2023), com o mesmo conjunto de dados composto por falhas de software que ocorreram em computadores executando o Win10 e Win11.

Como na maioria dos sistemas operacionais, o Windows possui seu próprio sistema de registro e gerenciamento de *logs*, arquivos estes que são gerados pelo SO contendo informações sobre as operações, as atividades e os padrões de uso de uma aplicação e serviços do sistema (AMAZON, 2023). Eles incluem um registro histórico de todos os processos, eventos e mensagens junto com dados descritivos adicionais, como carimbos de data/hora, para contextualizar essas informações. Dessa forma, se algo der errado com

os sistemas as causas podem ser investigadas a partir destas informações. Podendo então serem gerenciados pela aplicação *Windows Event Log* presente nos sistemas da Microsoft (MICROSOFT, 2012).

Na primeira versão do X-RAT, com o intuito de ter uma visão geral da confiabilidade de um sistema, apenas os eventos de falha foram filtrados e analisados, englobando tanto as falhas de SO, quanto as de aplicações do usuário.

Nos sistemas analisados, o serviço *Windows Event Log* armazena dados sobre eventos do sistema em arquivos de extensão *evtx* (BOTT; SIECHERT; STINSON, 2016). Portanto, a primeira etapa do trabalho (PEIXOTO, 2023) foi o desenvolvimento de funcionalidades para leitura desse tipo de arquivo. Seguindo com o desenvolvimento de um *parser*, para que fosse possível diferenciar os eventos de falha dos outros tipos de eventos armazenados nos arquivos de *log*. Finalizando esta parte dos eventos com o armazenamento em um banco de dados para que possam ser trabalhados a diante. Análises estatísticas sobre o conjunto de falhas de cada computador investigado foram realizadas. Consequentemente, foi desenvolvida em (PEIXOTO, 2023) uma funcionalidade capaz de automatizar os cálculos estatísticos e de estimar métricas de confiabilidade a partir dos dados de falha armazenados no banco de dados.

Diante disto, neste trabalho o foco se deu em acrescentar melhores representações dos dados de eventos de falha, através de novos gráficos apresentados no relatório de confiabilidade, utilizando a análise feita na primeira versão do software. A principal etapa foi a inclusão da previsão de falha de software com base em padrões de eventos múltiplos na aplicação X-RAT, que foi realizada no estudo feito por (SANTOS; JR; TRIVEDI, 2020b).

Por fim, os principais resultados obtidos por meio das análises descritas nas etapas anteriores são descritos em um relatório sobre a confiabilidade de software do computador analisado. Desta forma, foi produzida em (PEIXOTO, 2023) uma funcionalidade de geração automática do relatório para o usuário no formato PDF. Contendo as principais informações obtidas através dos *logs*, como o número de identificação do computador, sistema operacional que está sendo utilizado, quantidade de falhas, tipos de falhas, período do dia em que as falhas ocorreram, falhas mais recorrentes, gráficos de análise das falhas, categorização das falhas, métricas de confiabilidade e previsão de falhas de software.

1.4 Demais capítulos

Nos demais capítulos serão apresentados a revisão literária, a apresentação da plataforma X-RAT, os resultados obtidos após as análises, demonstrar o uso da plataforma e por fim a conclusão do trabalho.

No capítulo 2 é feita a revisão da literatura contendo uma base conceitual que visa definir com mais clareza os aspectos teóricos envolvidos no trabalho para uma melhor compreensão do tema.

No capítulo 3 é apresentada a visão geral sobre o que é o software X-RAT e todas as suas funcionalidades (banco de dados de mensagens de falha, parser, analisador e a previsão das falhas do SO), concluindo com o objetivo final que é a geração do relatório de confiabilidade.

No capítulo 4 são apresentados os resultados obtidos neste trabalho, incluindo um exemplo completo do uso da plataforma X-RAT e novas funcionalidades que podem ser desenvolvidas em futuros trabalhos.

O capítulo 5 contém as conclusões técnicas sobre o trabalho, o que foi aprendido durante sua execução e as principais dificuldades encontradas.

Ao final do documento são apresentadas as referências bibliográficas.

2 Revisão da Literatura

Neste capítulo, será apresentado o referencial teórico, contendo uma base conceitual que visa definir com mais clareza os aspectos teóricos envolvidos no trabalho e também os trabalhos relacionados para uma melhor compreensão do tema.

2.1 Fundamentação Teórica

Como base para um melhor entendimento do trabalho, alguns conceitos básicos sobre confiabilidade de software devem ser esclarecidos, seguindo a descrição em (AVIZIENIS et al., 2004).

- **Erro**: É o desvio do funcionamento correto. Erros podem ser transformados em outros erros (propagação de erro). A propagação de erro leva um sistema a falhar se um erro é propagado até a interface de serviço do sistema, ou seja, provocando o desvio do serviço em relação a sua especificação.

- **Falta**: a causa de um erro é a ativação de uma falta (defeito ou *bug*). Uma falta é inicialmente considerada inativa, e sua ativação causa um erro que pode atingir ou não a interface do serviço.

- **Falha**: Uma falha é a percepção de um erro no serviço fornecido.

- **Kernel**: núcleo do sistema operacional, que é a parte principal de um computador, responsável por conectar o software ao hardware.

Algumas definições das principais métricas de confiabilidade utilizadas neste trabalho serão apresentadas também, se baseando nos estudos de (LYU; NIKORA, 1992).

- **Confiabilidade (*reliability*)**: é a probabilidade que o sistema irá operar sem falha no intervalo de zero até o tempo t, sob determinadas condições operacionais.

$$R(t) = P(T > t), t \geq 0$$

onde T denota o tempo até a falha.

- **Probabilidade de falha - *U(t)* (*unreliability)***: é probabilidade que o sistema irá falhar até o tempo t, sob determinadas condições operacionais.

$$U(t) = P(T \leq t), t \geq 0$$

- **Função de risco - *h(t)* (*hazard rate*)**: é definida como h(t), sendo o limite da taxa de falha quando o intervalo (Δt) tende a zero.

$$h(t) = \frac{f(t)}{R(t)}$$

• **Tempo entre falhas - *TBF* (*time between failures*)**: é o tempo entre a ocorrência de uma falha e outra. Dado uma falha que ocorreu no tempo $t1$ e outra falha que ocorreu no tempo $t2$.

$$TBF = t1 - t2$$

• **Tempo médio entre falhas - *MTBF* (*mean time between failures*)**, é uma importante medida aplicada para sistemas reparáveis. Indica o tempo médio entre as falhas, é uma combinação das métricas $MTTR$ e $MTTF$.

$$MTBF = MTTR + MTTF$$

• **Tempo médio para reparo - *MTTR* (*mean time to repair*)**: é o valor esperado do tempo de reparo t, $E(t)$.

$$MTTR = \int_0^\infty tg(t)dt$$

• **Tempo médio para falha - *MTTF* (*mean time to failure*)**: É o valor esperado de t, $E(t)$, ou seja o tempo médio até a falha.

$$MTTF = \int_0^\infty tf(t)dt$$

• ***Warranty time***: é definido pelo maior TBF observado no conjunto de dados.

• **Disponibilidade A(t) (*availability*)**: é definida como a probabilidade que o sistema estará operando corretamente no tempo t.

$$A(t) = \frac{MTTF}{MTTF + MTTR}$$

onde MTTF, MTBF e MTTR foram obtidos considerando o tempo de missão t.

• **Taxa de falha**: Definido como a probabilidade de uma falha por unidade de tempo, ocorrer no intervalo [t1, t2], dado que não tenha ocorrido falhas antes de $t1$.

$$Taxa\ de\ falha = \frac{R(t1) - R(t2)}{(t2 - t1)R(t1)}$$

• **Função densidade de probabilidade (*probability density function*)**: define a função de probabilidade que representa a densidade de uma variável aleatória contínua situada entre um intervalo específico de tempo.

$$P(a < T < b) = \int_a^b f(t)dt$$

2.2 Trabalhos Relacionados

O quesito confiabilidade é um dos aspectos mais importantes na escolha de um produto. Quando o produto em questão é um software, a confiabilidade tem que ser ainda maior, e as pesquisas nesta área buscam aumentar a confiabilidade dos softwares que são desenvolvidos. Devido as aplicações dependerem do sistema operacional para serem executadas, falhas que ocorrem nos sistemas operacionais precisam também serem estudadas.

Com isso, o artigo (GANAPATHI; PATTERSON, 2005) apresenta uma análise das falhas de *kernel* do Windows XP. Apesar de ter sido realizada com um pequeno número de amostras, foi possível concluir que, o maior responsável pela maioria das falhas não é o SO, mas sim os *drivers* mal escritos dos dispositivos. Portanto, caso os usuários utilizem softwares bem desenvolvidos, reduzirá a quantidade de falhas durante a utilização.

Ao analisar a confiabilidade do Win7 no trabalho de mestrado (SANTOS; JR., 2017), é realizada uma análise de 7.007 registros de falhas reais obtidos de quatro ambientes de trabalho diferentes, que foram divididos em grupos de acordo com suas principais características para que possam ser analisadas. Ocasionando em 113 tipos de falhas de com base em seu ProductName, sendo 35 de OS Kernel, 23 de OS Application, e 55 de OS Service. Assim, qualquer falha poderia ser uma Falha de Referência, que seria utilizada como referência base para a busca por padrões de Sistemas Operacionais. Foram definidos também os Padrões de Falha de SO Candidatos, classificados como Eventos de Falhas Anteriores e Posteriores que ocorrem juntamente com a Falha de Referência. Com os Padrões de Falhas Candidatas, é possível definir o Padrão de Falha de SO, que acontece quando o mesmo PFC se repete em diferentes computadores. Com isso, foi definido um ranking para Padrão de Falha do SO, e assim, quanto mais bem qualificado o PFC, mais provável que ele seja um padrão de falha. Onde foram encontrados 45 padrões de falhas de SO, padrões que ocorreram em no mínimo 3 dos 4 grupos de computadores analisados na pesquisa. Formalizando então conceitos de divisão de falhas de SO, facilitando futuras análises baseadas nas qualificações apresentadas.

O trabalho de (SANTOS; JR, 2017) tem como objetivo identificar um padrão para os registros de falhas encontrados no Win7. Foi feita a caracterização das falhas presentes em computadores utilizados em diversos ambientes de trabalho, percebendo então que as falhas mais habituais consistiam em atualizações de software, devido a fatores como falta de espaço no disco e execução de serviços do SO concorrentes. O artigo (SANTOS; JR, 2016) visa entender melhor como funcionam as falhas, apresentando um estudo exploratório com falhas reais. Para isso, foram analisadas 7.007 falhas reais, divididas entre os grupos de aplicação, serviço e *kernel*, para que fosse possível realizar análises qualitativas e quantitativas. Por fim, foi possível concluir que as falhas de serviço são os mais comuns, e evidências confirmaram a presença de correlações nas falhas, e uma relação casual entre

diferentes máquinas.

Um dos últimos sistemas operacionais lançados pela Microsoft, o Win10 possui poucos estudos sobre sua confiabilidade. Em (OLIVEIRA, 2018), é realizado a análise de sua confiabilidade. Para isso, foi desenvolvido software capaz de extrair as informações contidas nos arquivos de falhas armazenados nas máquinas que utilizam este sistema. Após a extração foi utilizado a ferramenta OSRat (THE. . . , 2016). Desenvolvida pelo HPDCS (HIGH. . . , 2011) para analisar os dados. O trabalho consiste no desenvolvimento e estudo da interface gráfica e linguagem proposta para o desenvolvimento da aplicação. Posteriormente, foi feito a extração das informações dos arquivos de falhas, de modo a serem agrupadas e classificadas. Por fim, foi realizada a análise das falhas e viabilizar o estudo sobre um dos sistemas operacionais mais utilizadas atualmente.

Um aspecto fundamental na área de confiabilidade é identificar e compreender as causas e efeitos das falhas. No artigo (SANTOS; JR; TRIVEDI, 2021) foi realizado análises de dados de falha de software de campo, procurando caracterizar suas causas. As falhas analisadas foram coletadas de centenas de sistemas de computadores localizados em diferentes locais de trabalho. Foi considerado diferentes aspectos de cada causa de falha analisada, como seu tipo, contexto, camada de software e código onde ela se manifestou. Descobrindo então que 84% das causas de falha estavam relacionadas ao endereçamento de memória, capacidade de resposta e tratamento de exceções. A taxa das causas de falha predominantes parece correlacionar-se com a duração do tempo de execução dos códigos com falha. Independentemente do tipo de código com falha, as causas de falha predominantes estavam relacionadas à programação. Estudos empíricos mostraram evidências robustas de padrões de falha do sistema operacional caracterizados por múltiplas combinações de eventos de falha compostos pelos mesmos ou diferentes tipos de falha.

No artigo (SANTOS; JR; TRIVEDI, 2020a), foi apresentada uma abordagem estatística para prever falhas de sistema operacional com base na associação de múltiplas falhas. Uma vez que identificamos associações sistemáticas de falhas nos dados de campo, calculamos a probabilidade de uma determinada falha ocorrer dentro de um intervalo de tempo após a ocorrência de um padrão específico de falhas anteriores. Devido à natureza dos dados de falha, em que os tipos de falha devem ser tratados como variáveis categóricas, utilizando o método de regressão logística para abordar o problema de pesquisa. Esta abordagem foi capaz de prever falhas do sistema operacional com uma boa precisão de (81% a 95%). Os modelos de regressão resultantes se mostraram robustos o suficiente para lidar com diferentes intervalos de tempo de previsão sem nenhum efeito degradante em sua precisão.

Por fim, o estudo mais recente feito também por um membro do grupo .SDLAB também foi realizado no Win10, se estendendo para o Win11 em (PEIXOTO, 2023), que desenvolve uma ferramenta para análise de confiabilidade de software, que realiza a coleta,

parsing e análise de *logs* de falhas, produzindo um relatório de confiabilidade contendo as analises textuais e gráficas para o usuário em formato PDF. Por ser um trabalho recente e do grupo .SDLAB é a partir dele que daremos continuidade na evolução do software de análise de falhas em questão.

3 Plataforma X-RAT

Nesta seção, são apresentados os aspectos gerais presentes na plataforma X-RAT, descrevendo todas suas funcionalidades e o seu objetivo final. Nas seções 3.3 a 3.3.3 serão descritos os módulos desenvolvidos na primeira versão do relatório de confiabilidade e na seção 3.3.4 será apresentada as funcionalidades desenvolvidas neste trabalho.

3.1 Visão Geral

O grupo .SDLAB foi o responsável pelo desenvolvimento da plataforma de análise de confiabilidade de software X-RAT, que possui o intuito de avaliar a confiabilidade de sistemas de forma automatizada. Disponível para todos que desejam verificar a confiabilidade de seus sistemas, incluindo usuários domésticos, empresas e organizações.

A plataforma consiste em uma ferramenta capaz de analisar *logs* de falhas de diferentes aplicações de software, possuindo um sistema coletor de dados que recebe os arquivos de *log*, os quais são analisados através da *engine*, com o objetivo de no final do processo gerar um relatório de confiabilidade para o software em questão.

3.2 Relatórios de Confiabilidade

O objetivo principal do X-RAT é gerar os relatórios de confiabilidade para o usuário final, que são documentos que têm como objetivo fornecer informações ao usuário a respeito da confiabilidade do software analisado. Relatórios estes que têm sua importância por permitirem tanto aos usuários casuais, quanto desenvolvedores e provedores de serviço, a entender como as aplicações estão funcionando em um sistema real a fim de identificar qualquer problema de confiabilidade que possa acarretar mal funcionamento da aplicação, facilitando tomadas de decisões em possíveis necessidades de correções.

Será apresentado adiante, no cenário de uso, um exemplo de um relatório de confiabilidade, no qual após o envio dos arquivos de *log* ao X-RAT, é realizado *parsing* e análise de *logs* de falhas, e por fim é produzido um relatório de confiabilidade descrevendo as aplicações onde mais ocorrem falhas, o período mais propício de ocorrer, gráficos demonstrativos e a previsão de ocorrência das falhas.

3.3 Projeto Desenvolvido

O principal objetivo deste trabalho é a continuação do desenvolvimento da plataforma de análise de confiabilidade (X-RAT), mas se faz necessário a apresentação das etapas realizadas na primeira versão da plataforma (PEIXOTO, 2023) para um completo entendimento de seu funcionamento.

O escopo da *engine* de análise é composto por:

- Coleta dos dados;
- Filtragem das falhas;
- Classificação e análise das falhas;
- Estimação das métricas de confiabilidade;
- Geração do relatório de confiabilidade.

Em (PEIXOTO, 2023), detalhes da *engine* são explicitados, onde dois módulos principais dividem sua arquitetura. O módulo *parser* é responsável por efetuar a leitura dos eventos de falha presentes no arquivo em formato *evtx*, padronizar os dados, encontrar a mensagem de falha referente ao evento, classificar o tipo de falha e armazenar os eventos em estrutura previamente definida no banco de dados.

O modulo Analisador é responsável por calcular as métricas baseadas na data e horário de ocorrência das falhas, que também calcula o *TBF* das falhas. Ele também é responsável por realizar testes de aderência sobre uma lista de distribuições de probabilidade candidatas, a fim de determinar qual distribuição melhor se adere ao conjunto de dados de falha. E por fim gera o relatório de confiabilidade, em formato PDF, contendo informações sobre o conjunto de dados analisado, as métricas estimadas, falhas mais recorrentes, o ranking de distribuições, um gráfico de densidade das falhas por minutos e o *plot* de gráficos.

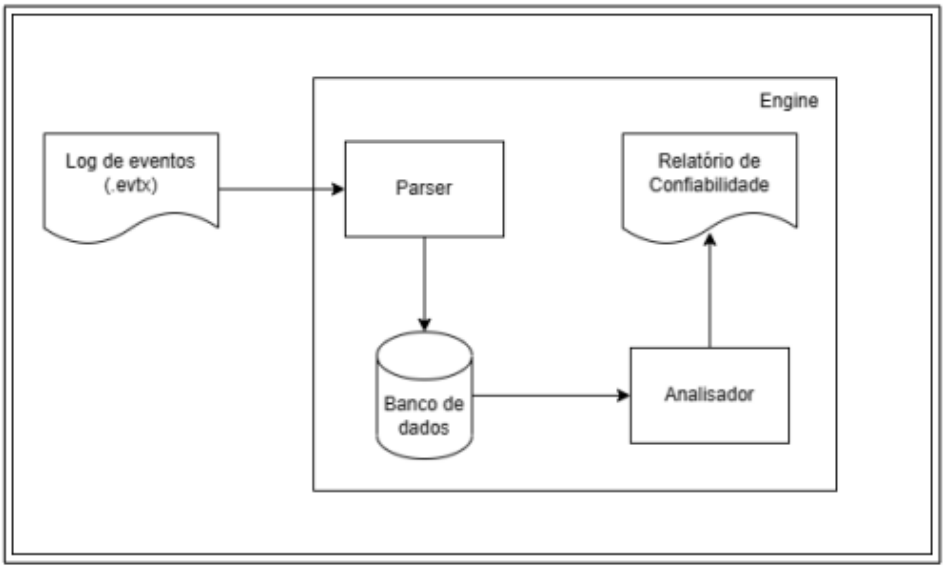


Figura 1 – Arquitetura da Engine de análise.

Fonte: (PEIXOTO, 2023)

3.3.1 Banco de dados de mensagens de falha

A primeira versão do Relatório desenvolvida por (PEIXOTO, 2023), foi construída uma base de dados de mensagens de falha, visto que estas não são armazenadas nos *logs* de falhas, mas sim obtidas por meio de consulta em tempo real ao registro da aplicação que publicou o evento de falha.

No SO Microsoft Windows existe o software Event Viewer, que é responsável por mostrar a mensagem de falha, de acordo com a origem da falha, como mostrado na Figura 2.

Figura 2 – Event Viewer.

Este software utiliza dos campos Provider (nome da aplicação) e EventID (código de cada evento) como identificadores do evento e realiza a consulta no registro da aplicação pela mensagem referente ao evento desejado. O campo EventData contém informações específicas do evento de falha.

Para a construção de uma base de dados de mensagens de falhas, foi necessária a utilização de um software chamado Event Sentry (NETIKUS, 2022), que mapeia todas as aplicações disponíveis no computador local e gera um arquivo no formato *txt* contendo mensagens de falha presentes em cada aplicação individual.

A partir da obtenção dos arquivos contendo o mapeamento das mensagens de falha por aplicação (Provider), foi desenvolvido um algoritmo em Python responsável pela leitura, *parsing* e armazenamento das mensagens no banco de dados.

3.3.2 Parser

A extensão de arquivos *evtx* é um formato desenvolvido pela Microsoft, para que os eventos de *log* sejam gravados no arquivo em formato *xml* e codificados em binário com intuito de serem posteriormente decodificados pela aplicação Event Viewer quando requisitada leitura. Por este motivo foi necessária a implementação de um módulo capaz de decodificar os arquivos *evtx*, realizar o *parsing* dos campos *xml* e aplicar algoritmos de tratamento de dados para que os *logs* possam ser visualizados corretamente.

O desenvolvimento do Parser realizado por (PEIXOTO, 2023) é composto por 5 etapas:

Primeira etapa:

- Realização da decodificação do arquivo *evtx* gravado em formato binário para *xml*. Para tal, foi utilizada a biblioteca do Python pyevtx-rs (BENAMRAM, 2022), que realiza a decodificação do arquivo e retorna um vetor iterável onde cada elemento é um evento representado em formato *xml*, como descreve a Figura 3.

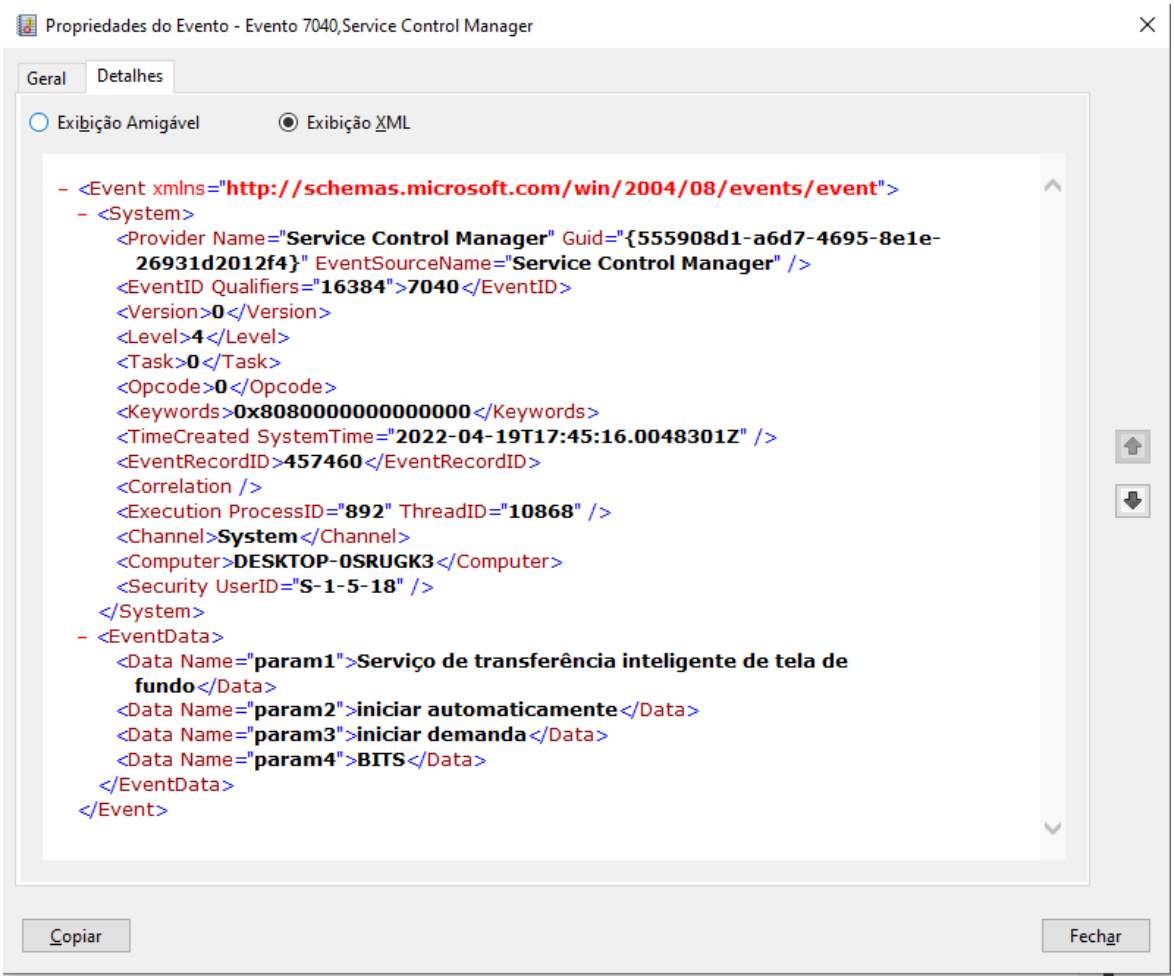


Figura 3 – Estrutura de um evento do Win10 no arquivo de log exibido em xml.

Segunda etapa:

- Implementação de um algoritmo que realiza a extração das informações de cada um dos campos do evento da estrutura *xml* correspondente. Nesta etapa foi a utilizada a biblioteca *lxml* (TEAM, 2022), que fornece uma interface para leitura e extração de textos *xml*, sendo cada evento de *log* do vetor iterável processado individualmente em um *loop*.

Terceira etapa:

- Armazenamento em memória dos eventos de falha extraídos na etapa anterior, em formato *dataframe*, através da biblioteca Pandas (TEAM, 2022). Esta biblioteca é

responsável por fornecer uma interface para operações em grandes conjuntos de dados com boa performance, a qual será utilizado ao longo de todo o desenvolvimento do software.

Um dos problemas encontrados ao longo do desenvolvimento do parser se deu pelo fato de que a Microsoft converte o horário do evento do *log* para o *timezone* UTC, perdendo-se o horário local em que ocorreu o evento. Porém é possível encontrar qual o *timezone* do computador do usuário por meio de um evento informativo que armazena esta informação, sendo este evento representado pelo campo EventID número 6013.

Quarta etapa:

- Busca e extração do campo com EventID igual a 6013, para encontrar o *timezone* do computador em que ocorrem as falhas. Criando então um novo campo Time, que consiste na data e horário disponíveis no campo TimeCreated com a aplicação do *timezone* encontrado, para obter o horário correto da falha.

Por último, as seguintes operações são realizadas nesta etapa de *parsing*:

- É realizada uma consulta no banco de dados pelas mensagens de falha, que posteriormente são mapeadas aos eventos que estão sendo processados em um novo campo Message.

- É aplicado um algoritmo que realiza a classificação das falhas entre as seguintes categorias (JR; OLIVEIRA; ARAUJO, 2013):

Aplicação de usuário (USER_{APP}): Contém falhas em aplicações de usuário, que são programas com os quais os usuários interagem diretamente (Exemplo: notepad.exe).

Aplicação de SO (SO_{APP}): Esta categoria contém falhas causadas pelo mau funcionamento de aplicativos do sistema operacional, os quais são executados sob demanda do usuário (Exemplo: explorer.exe).

Serviço de SO (SO_{SVC}): Inclui falhas de serviços do sistema operacional que, geralmente, executam em segundo plano e com mínima ou nenhuma interação do usuário (Exemplo: MsInstaller).

Kernel (SO_{KNL}): A última categoria possui falhas causadas pelos subsistemas do *kernel* do SO, as quais exigem a reinicialização do sistema na maioria dos casos (Exemplo: microsoft-windows-driverframeworks-usermode, que indica que um problema em um ou mais *drivers* de modo de usuário, finalizando o processo de *host*).

- E por fim, os eventos de falha são armazenados no banco de dados.

3.3.3 Analisador

O Analisador desenvolvido por (PEIXOTO, 2023), representa a fase final da *engine* de análise, seu código fonte foi implementado na linguagem Python e utiliza da biblioteca

rpy2 (GAUTIER, 2022), que fornece uma interface para executar algoritmos escritos na linguagem R embutidos no código fonte Python, que tem como diferencial a possibilidade de realizar análises estatísticas dos dados obtidos.

Assim como o Parser, o Analisador foi dividido em 5 etapas, sendo:

Primeira etapa:

- Leitura dos eventos do *log* no banco de dados.
- Armazenamento dos eventos em um *dataframe* do Pandas (TEAM, 2022).
- Realização das operações sobre os dados (reúne informações úteis sobre o *dataset*, como: ID do computador em que ocorreram as falhas, primeira e última data de ocorrência de um evento, sistema operacional e tipos de eventos).

Segunda etapa:

- Identificação e separação dos eventos com base no campo Level. Os eventos com sucesso são representados pelo valor 0, aqueles que possuem valor igual a 1 e 2 são eventos de falha, eventos *warning* possuem valor 3, e por fim os eventos informacionais que são representados pelo valor 4.
- Separação das falhas em dois grupos, a partir dos tipos de falha obtidos durante a etapa de classificação (JR; OLIVEIRA; ARAUJO, 2013). O primeiro grupo consiste nas falhas de Aplicação de usuário (USER_{APP}) e o segundo nas falhas de Aplicação de SO (SO_{APP}), Serviço de SO (SO_{SVC}) e Kernel (SO_{KNL}).

Em seguida são calculadas as seguintes métricas de confiabilidade:

- Quantidade e porcentagem do número total de falhas por grupo.
- Falhas por horas do dia de cada grupo e número de falhas por dia da semana de cada grupo.
- Análise para mapear qual a fonte das falhas mais recorrentes de cada grupo.

Terceira etapa:

- Cálculo dos *TBFs* do conjunto de falhas, e aplicação do método Curve Fitting para as distribuições de probabilidade candidatas Normal, Lognormal, Weibull, Exponencial e Gamma. Com isso, são realizados os testes de aderência Anderson-Darling (CULLEN, 1999), Kolmogorov-Smirnov (CULLEN, 1999) e Akaike information criterion(AKAIKE, 1974) sobre as distribuições obtidas.
- Construção de um ranking com os valores referentes a cada uma das distribuições candidatas, utilizando os resultados obtidos nos testes de aderência, com o intuito de se escolher a distribuição que melhor se adéqua ao conjunto de dados (*TBFs*). A distribuição escolhida é utilizada para o cálculo das métricas de confiabilidade, sendo neste trabalho

calculadas as métricas *MTBF*, *Warranty time*, R(t), h(t), com t variando em diferentes tempos de missão em horas.

Quarta etapa:

- Exportação do relatório de confiabilidade em formato PDF, contendo todas as informações obtidas nas etapas anteriores, junto com o histograma de densidade dos *TBFs*, e o gráficos referentes a distribuição escolhida: histograma de densidades empírica e teórica, Q-Q plot e P-P plot, utilizando a biblioteca reportlab (ROBINSON, 2022), disponível para a linguagem Python, que permite desenhar formas, plotar gráficos e exportar arquivos em formato PDF.

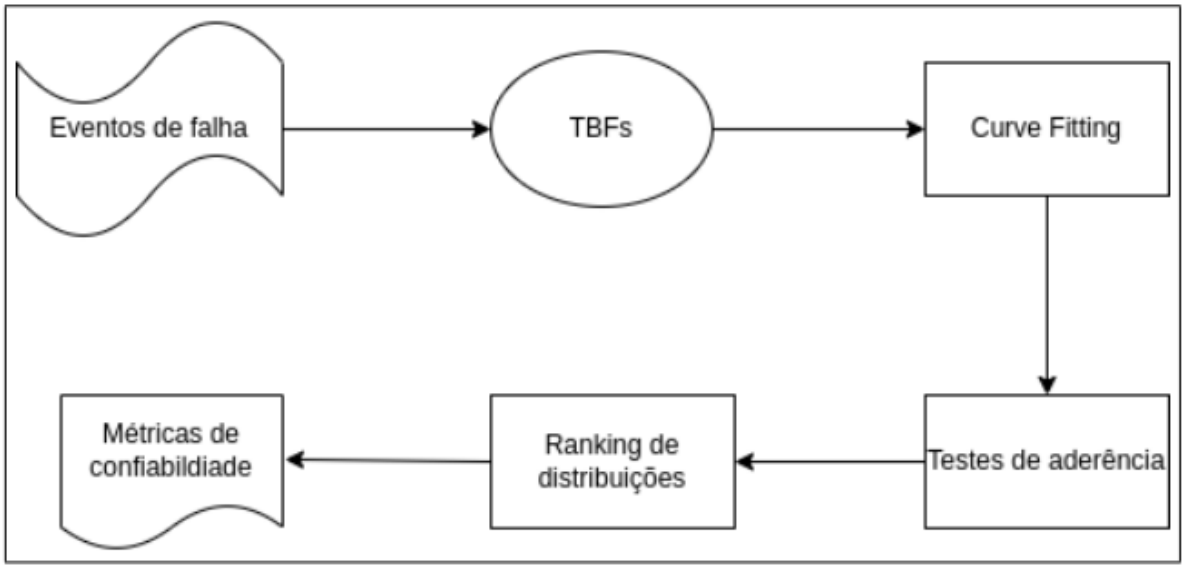


Figura 4 – Etapas do Analisador.

Fonte: (PEIXOTO, 2023)

3.3.4 Previsão das falhas do SO com base nas Associação de Falhas

A previsão de falhas do SO é a nova análise adicionada na plataforma X-RAT, a partir de um estudo feito por (SANTOS; JR; TRIVEDI, 2020a), que apresenta uma abordagem estatística para prever falhas de sistema operacional com base na associação de múltiplas falhas. Visto que foi identificado associações sistemáticas de falhas nos dados de campo, é realizado o cálculo de probabilidade de uma determinada falha ocorrer dentro de um intervalo de tempo após a ocorrência de um padrão específico de falhas anteriores.

Existe uma base de conhecimento que é usada para estabelecer os padrões de falhas, esses padrões de falhas são comparados com os padrões encontrados na amostra enviada pelo usuário. Esta base de conhecimento é ampliada a partir dos *logs* recebidos pelos usuários. Com isso, é utilizado o método de regressão logística multinomial para resolver

o problema proposto, por possuir um bom desempenho, como ocorreu em (SANTOS; JR; TRIVEDI, 2020a), com uma porcentagem de previsão de falhas do SO de (81% a 95%).

Neste trabalho foi considerado para a análise, as falhas do sistema operacional, utilizando a mesma abordagem que foi utilizada no Parser apresentada na Seção 3.3.2, que considera as quatro categorias de falhas, [SO_{APP}], [SO_{SVC}], [SO_{KNL}] e [$USER_{APP}$].

Segundo (SANTOS; JR; TRIVEDI, 2020a), uma associação de falha do sistema operacional é considerada uma relação sistemática entre uma falha do sistema operacional com outra falha anterior do SO.

Os dados utilizados para análise neste do trabalho foram coletados de diferentes computadores. São compostos por registros de falhas de SO coletados de *logs* de máquinas com o Win10 e Win11. Neste conjunto de dados citado acima, possui 530.633 falhas de sistema operacional, que foram divididos em quatro grupos (G1 a G4). O grupo G1 possui 124.465 falhas, o G2 338.063, G3 44.010 e por fim o G4 com 24.095.

Para procurar associações de falhas genuínas, primeiro é calculado o tempo entre as falhas do sistema operacional usando as etapas ilustradas na Figura 4.

Filtramos então o conjunto de dados para selecionar apenas computadores com mais de uma falha. Para cada evento de falha da amostra de trabalho, é criado os campos (DeltaTime_Imm_Before e Categoria_Imm_Before) na *database* como candidatos à associação de padrões de falhas, para armazenar a categoria e seu tipo de falha anterior e também o Δt em relação a ele, como mostra a figura a seguir.

	Computername	TimeCreated	Categoria	DeltaTime_Imm_Before	Categoria_Imm_Before
0	001_G1	2016-01-15 17:26:40.316735+00:00	S	NaN	NaN
1	001_G1	2016-01-15 17:31:51.657147+00:00	S	8.648345e-02	S
2	001_G1	2016-01-15 19:01:01.765724+00:00	S	1.486141e+00	S
3	001_G1	2016-01-15 19:01:01.766193+00:00	S	1.302778e-07	S
4	001_G1	2016-01-15 19:01:07.562849+00:00	S	1.610182e-03	S

Figura 5 – Database com falhas candidatas.

Para cada *log* do sistema na amostra de trabalho, o código desenvolvido por (SANTOS; JR; TRIVEDI, 2020a) encontra seu último registro de falha do sistema operacional e, em seguida, verifica se a falha anterior do sistema operacional ocorreu no mesmo dia ou no dia anterior.

Depois calculamos o Δt entre a última falha e a penúltima falha. Armazenamos a categoria e o tipo da penúltimo falha e o Δt entre eles nos três campos criados acima ((DeltaTime_Imm_Before e Categoria_Imm_Before) na *Database*).

Caso contrário, passamos para a falha anterior e repetimos as mesmas etapas. Por

fim, removemos todos os registros de falha sem valor nos três campos acima mencionados do banco de dados, ou seja, todos os registros de falha sem eventos de falha anteriores que ocorreram no mesmo dia ou no dia anterior são removidos.

Portanto, consideramos como uma associação de falha do SO toda combinação entre uma falha do SO e sua falha anterior com o Δt menor ou igual o valor do *Upper bound* encontrado para aquele tipo de associação, o que garante que a associação seja consistente dentro desse intervalo de tempo. Além disso, como estamos interessados em encontrar padrões sistemáticos, analisamos apenas as falhas recorrentes do SO que foram observados em, pelo menos, três dos quatro grupos da nossa amostra de trabalho, procurando uma elevada consistência dos resultados. A Figura: 6 exibe o fluxo de execução deste protocolo.

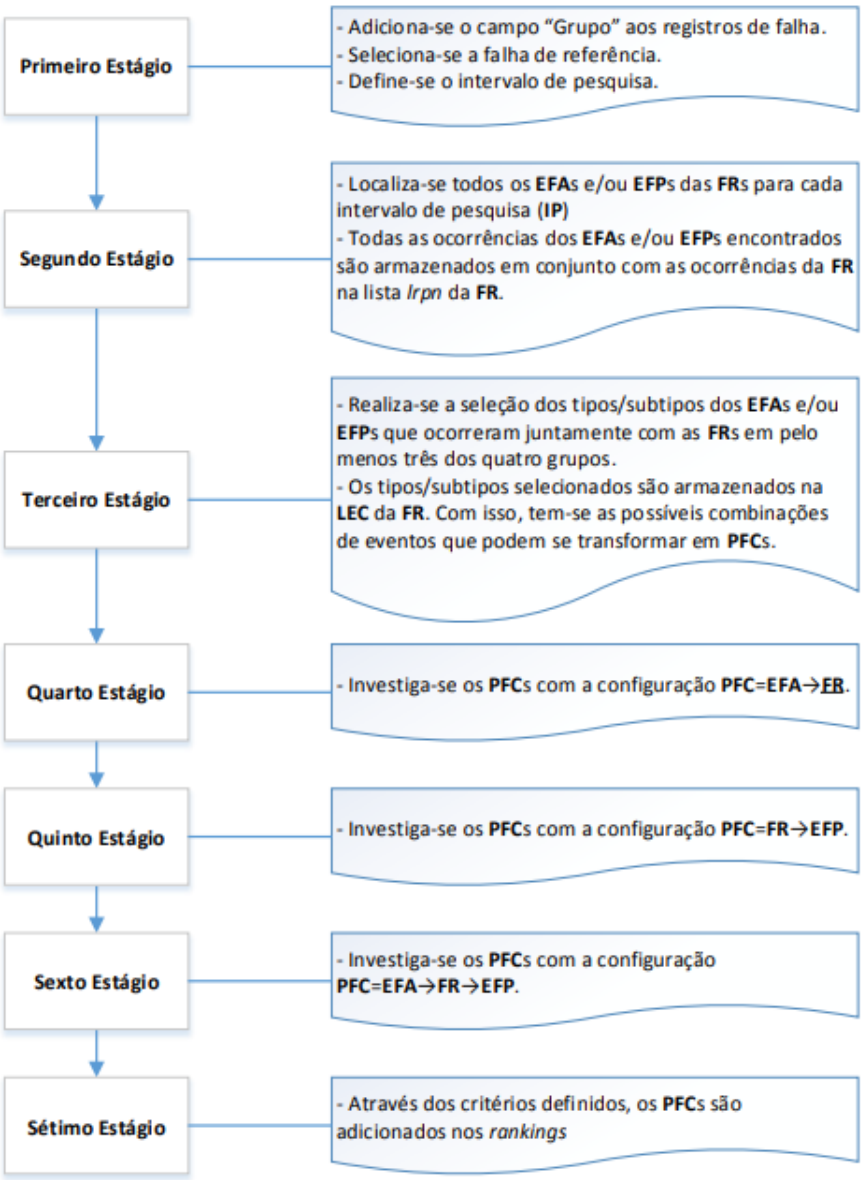


Figura 6 – Fluxo de execução do protocolo.

Fonte: (SANTOS; JR., 2017)

Tabela 1 – Predição baseada em associações de categorias de falhas do SO.

Observed Failure Category	**Probability**	**Probable Failure Category**
S	96.70	S
K	92.70	K
A	92.01	A
U	59.22	U
U	27.88	S
U	9.10	A
A	6.94	S
K	6.83	S
U	3.80	K
S	2.05	K
S	0.90	A
A	0.76	U
S	0.35	U
A	0.29	K
K	0.25	U
K	0.21	A

A Tabela 1 mostra as probabilidades de todos os valores das variáveis dependentes (Observed Failure Category - categorias de falha do próximo evento de falha) com base nos valores da variável independente (Probable Failure Category - categorias de falha do evento de falha anterior).

Notamos que a probabilidade é maior quando ambas as variáveis dependentes e independentes têm a mesma categoria de falha, especialmente para falhas [SO_{SVC}], com 96.7%. Essa descoberta indica que é mais provável que uma falha do sistema operacional ocorra junto com outras falhas do sistema operacional da mesma categoria de falha.

Seguindo a análise feita de que quando as variáveis dependentes e independentes possuem a mesma categoria de falha, observamos uma probabilidade alta de falha também entre [SO_{KNL}] com 92.7%, [SO_{APP}] com 92.0% e [USER_{APP}] com 59.2%.

4 Resultados

Neste capítulo é apresentado um exemplo completo do uso da plataforma X-RAT, as contribuições que este trabalho trouxe para a *engine*, e novas funcionalidades que podem ser desenvolvidas em futuros trabalhos.

4.1 Cenário de Uso

Nesta seção é descrito um cenário de uso completo da plataforma X-RAT. O primeiro passo é acessar o *website* da plataforma [1] para poder realizar o *upload* dos arquivos de *log*, necessitando apenas selecionar os *logs* na pasta descrita no *website*.

Segue imagem demonstrativa desta forma de realizar o *upload*:

Figura 7 – Interface para coleta de *logs* de falha.

Após o usuário selecionar os arquivos de *log* de seu computador, na tela seguinte o usuário responde breves perguntas a respeito de seu perfil de uso da máquina.

[1] <https://x-rat.facom.ufu.br>

Figura 8 – Questionário sobre o perfil do usuário.

Após o envio dos dados para o servidor da aplicação, a *engine* de análise será acionada e realizará o processamento dos *logs*, retornando para o usuário tanto as análises realizadas na primeira versão do X-RAT (PEIXOTO, 2023), quanto da versão atual a fim de ao final do processo, retornar para o usuário o relatório de confiabilidade em formato PDF, conforme é mostrado nas Figuras 9 e 10.

Software Dependability Laboratory

(sdlab.facom.ufu.br)

Reliability Analysis Report

System identification:

Computer ID: desktop-j7gvk15
Operating System: Windows 10

Dataset:

Failure types: Operating system and Applications
Sample period: 07/26/2021 - 04/05/2022

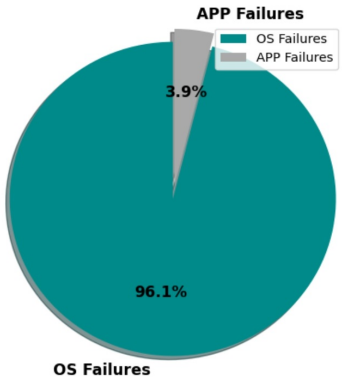
Number of failures:

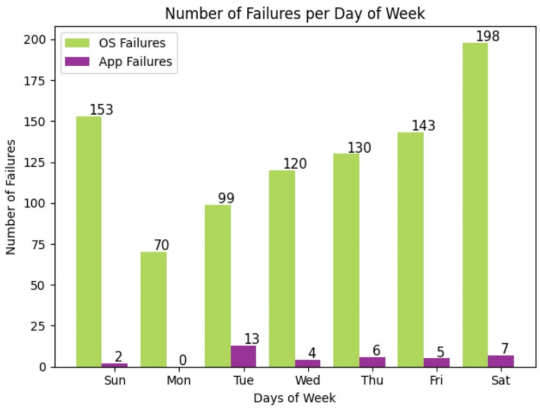
Operating System 913 (96.1%) | Applications 37 (3.9%)

Number of failures per Parts of Day:
12 am to 06 am: OS (19.3%) App. (24.3%)
06 am to 12 pm: OS (7.1%) App. (2.7%)
12 pm to 06 pm: OS (40.5%) App. (27.0%)
06 pm to 12 am: OS (33.1%) App. (45.9%)

Number of failures per Days of Week:

Sun: OS (16.8%) App. (5.4%)
Mon: OS (7.7%) App. (0.0%)
Tue: OS (10.8%) App. (35.1%)
Wed: OS (13.1%) App. (10.8%)
Thu: OS (14.2%) App. (16.2%)
Fri: OS (15.7%) App. (5.4%)
Sat: OS (21.7%) App. (18.9%)

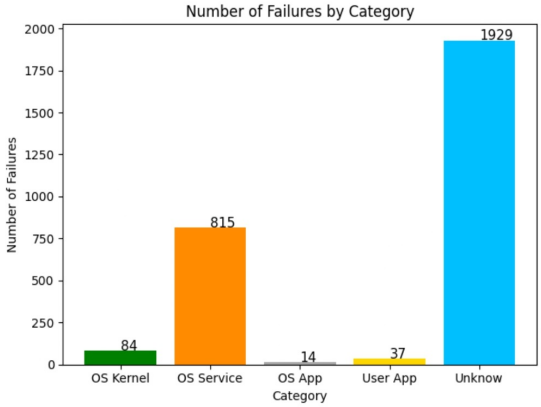




Most recurrent failures (OS vs APP):

OS
- service control manager
- microsoft-windows-ndis
- vss
- volsnap
- volmgr
- microsoft-windows-windowsupdateclient
- application error
- schannel
- microsoft-windows-wer-systemerrorreporting
- microsoft-windows-driverframeworks-usermode

APP
- application error
- application hang



Failure Categories

OS App: OS Application Process Failures;

OS Service: OS Service Component Failures;

OS Kernel: OS Kernel Subsystem Failures;

User App: Application Failed;

Unknow: Uncategorized Failures.

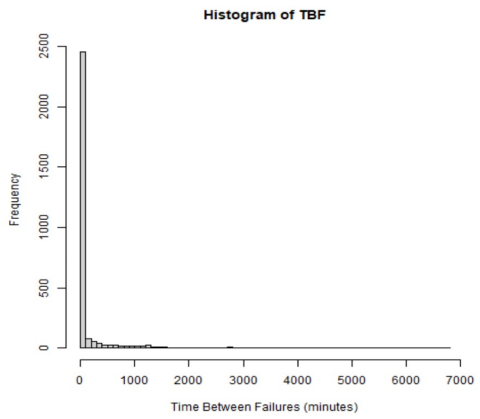
Figura 9 – Relatório de confiabilidade - Pág. 1

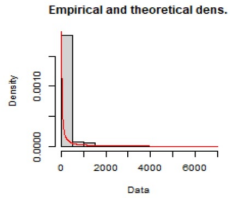
Software Dependability Laboratory

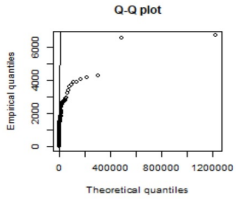
(sdlab.facom.ufu.br)

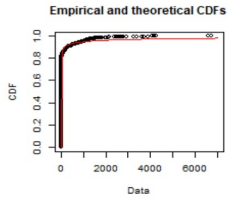
Reliability Analysis Report

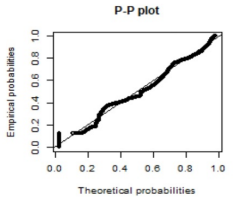
	Distribution	Params	Values	AIC	KS	AD	AIC_rank	KS_rank	AD_rank
1	weibull	scale, shape	0.62897288372756, 0.149069255256104	-5996.761602	0.095960	36.892442	1	1	1
2	gamma	shape, rate	0.0823340929688964, 0.000650800548011519	-5980.155666	0.119655	48.113322	2	2	2
3	lnorm	meanlog, sdlog	-4.55600611669301, 9.24326301721107	-5252.221599	0.140261	115.926296	3	3	3
4	exp	rate	0.00790438718928634	33618.981705	0.726940	nan	4	5	5
5	norm	mean, sd	126.512021571683, 440.901871993335	43218.671576	0.402211	nan	5	4	5











MTBF: 126.5 minutes

Warranty time: 6713.0 minutes

	h(t)	R(t)
1 hours:	h(1) = 0.455833326	R(1) = 0.139065399
2 hours:	h(2) = 0.352464825	R(2) = 0.112190263
4 hours:	h(4) = 0.272537013	R(4) = 0.088417679
8 hours:	h(8) = 0.210734286	R(8) = 0.067899380
12 hours:	h(12) = 0.181301154	R(12) = 0.057422697
24 hours:	h(24) = 0.140187818	R(24) = 0.042073150

Reliability Metrics

MTBF: Mean time between failures;

Warranty time: is defined by the largest TBF observed in the dataset;

h(t): is the probability of the first and only failure in a certain time t;

R(t): is the probability that the system will operate without failure in the range zero to time t, under certain operating conditions.

Observed Failure Category	Probability	Probable Failure Category
S	96.70	S
K	92.70	K
A	92.01	A
U	59.22	U
U	27.88	S
U	9.10	A
A	6.94	S
K	6.83	S
U	3.80	K
S	2.05	K
S	0.90	A
A	0.76	U
S	0.35	U
A	0.29	K
K	0.25	U
K	0.21	A

Predict Operating System Failures

Highest probability when both dependent variables and independent are of the same fault category, indicating which is most likely an operating system crash occurs along with other operating system failures same fault category, regardless of category.

Probabilities

[SOSVC] + [SOSVC] = 96.7%

[SOKNL] + [SOKNL] = 92.70%

[SOAPP] + [SOAPP] = 92.01%

[USERAPP] + [USERAPP] = 59.22%

Figura 10 – Relatório de confiabilidade - Pág. 2

4.2 Resultados Alcançados

Este trabalho teve como objetivo realizar o desenvolvimento da versão 2 da aplicação X-RAT, como resultado foi implementada duas novas análises possíveis de serem realizadas na *engine*.

A primeira nova funcionalidade adicionada a plataforma X-RAT foi a apresentação novos gráficos para melhor entendimento dos dados. Como mostrado na Figura 11, o primeiro gráfico ilustra a quantidade de falhas de sistema operacional em comparação com as falhas de aplicação.

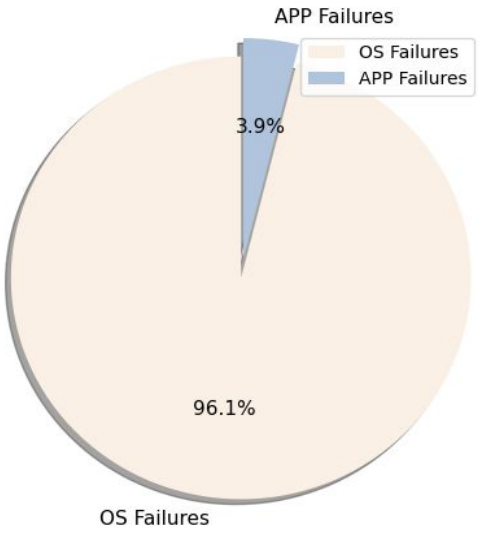


Figura 11 – Falhas de SO *vs* Falhas de Aplicação

Outro gráfico adicionado foi o demonstrativo de número de falhas por dias da semana, que apresenta em qual dia da semana ocorre a maior quantidade de falhas, Figura 12.

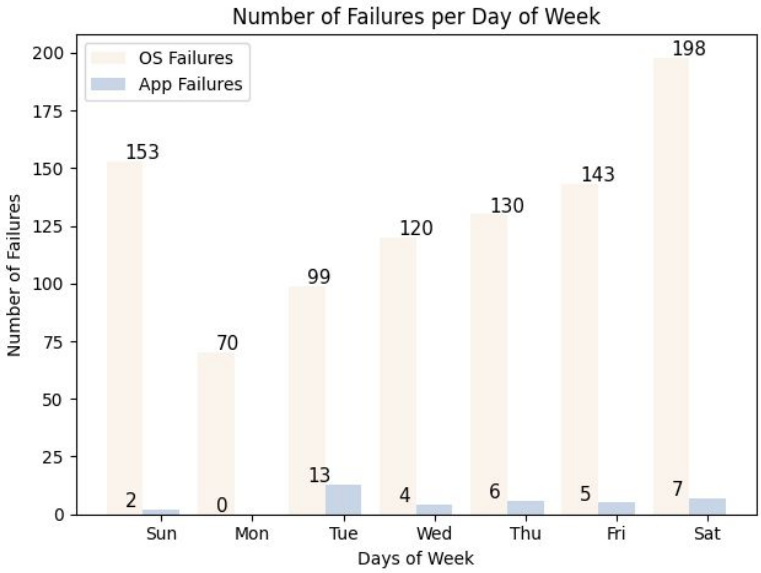


Figura 12 – Número de falhas por dias da semana.

A seguir na Figura 13, temos também o gráfico que demonstra a quantidade de falhas agrupadas por categorias, bem como uma breve descrição da diferença entre as categorias de falhas.

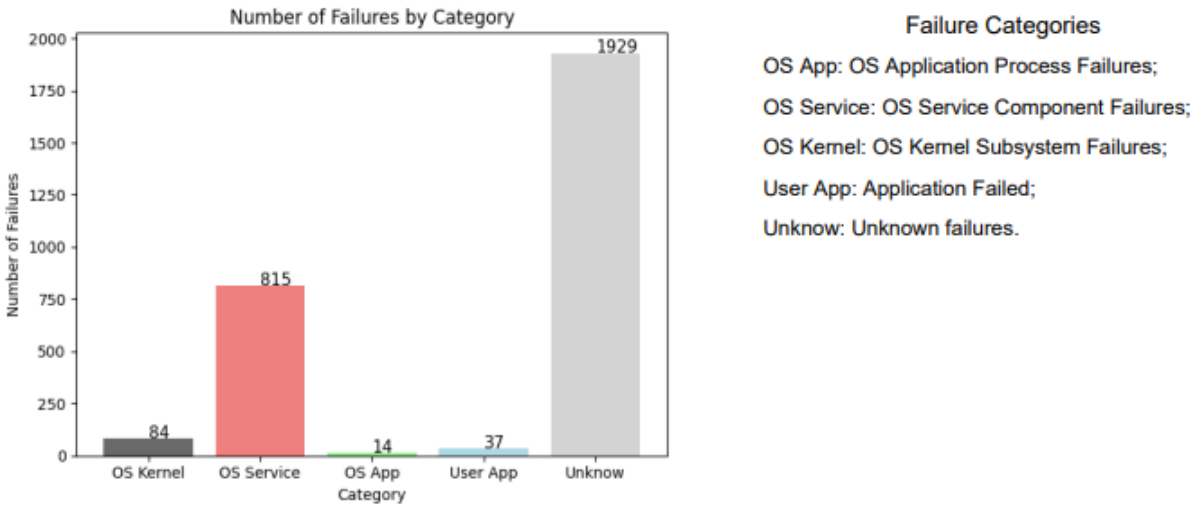
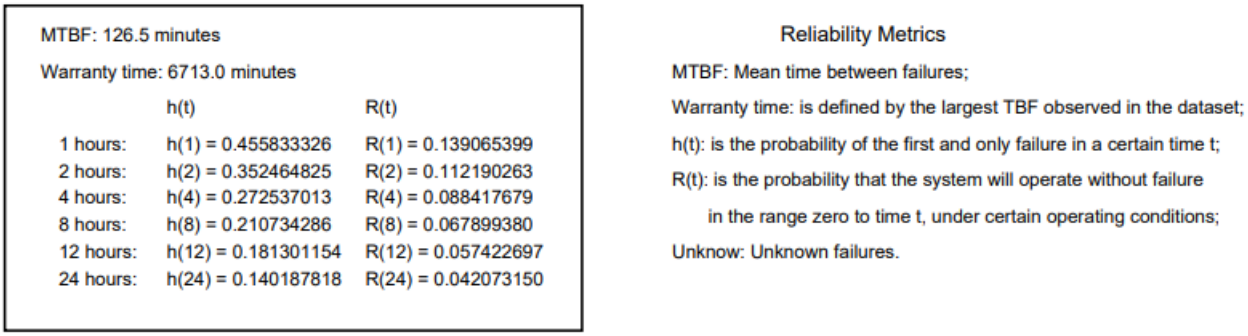


Figura 13 – Categorias mais recorrentes das falhas

Foi acrescentando também uma breve descrição das Métricas de Confiabilidade, descrevendo a diferença entre as métricas para melhor compreensão do usuário final na Figura 14.



MTBF: 126.5 minutes

Warranty time: 6713.0 minutes

	h(t)	R(t)
1 hours:	h(1) = 0.455833326	R(1) = 0.139065399
2 hours:	h(2) = 0.352464825	R(2) = 0.112190263
4 hours:	h(4) = 0.272537013	R(4) = 0.088417679
8 hours:	h(8) = 0.210734286	R(8) = 0.067899380
12 hours:	h(12) = 0.181301154	R(12) = 0.057422697
24 hours:	h(24) = 0.140187818	R(24) = 0.042073150

Reliability Metrics

MTBF: Mean time between failures;

Warranty time: is defined by the largest TBF observed in the dataset;

h(t): is the probability of the first and only failure in a certain time t;

R(t): is the probability that the system will operate without failure in the range zero to time t, under certain operating conditions;

Unknow: Unknown failures.

Figura 14 – Métricas de Confiabilidade

E por fim, a Figura 15 mostra a adição da funcionalidade de previsão de falhas do sistema operacional com base na associação de falhas, demonstrando a previsão de falhas referente ao *database* apresentado.

Observed Failure Category	Probability	Probable Failure Category
S	96.70	S
K	92.70	K
A	92.01	A
U	59.22	U
U	27.88	S
U	9.10	A
A	6.94	S
K	6.83	S
U	3.80	K
S	2.05	K
S	0.90	A
A	0.76	U
S	0.35	U
A	0.29	K
K	0.25	U
K	0.21	A

Figura 15 – Previsão de falhas

4.3 Trabalhos Futuros

Obtivemos um bom resultado após a conclusão deste trabalho com a geração do relatório em PDF, porém a plataforma X-RAT ainda possui diversas funcionalidades a serem desenvolvidas e aprimoradas.

Acrescentar também a previsão de falhas do SO de acordo com o tipo da falha, conforme é realizado em (SANTOS; JR., 2017) que pode ser um bom foco para um futuro trabalho.

Outro ponto que pode ser explorado, é a análise em conjunto de grupos de trabalho, apresentando os resultados de um laboratório, organização, bibliotecas.

Também é possível para trabalhos futuros atuar no contexto do coletor de dados para adicionar suporte a novos tipos de *logs* de falha, tanto para alguma aplicação em específico, quanto para outro sistemas operacionais, dado que inicialmente apenas são suportados *logs* de falha dos sistemas operacionais Win10 e Win11.

5 Conclusão

5.1 Técnica

Na primeira versão do relatório (PEIXOTO, 2023), foi utilizada a biblioteca pyevtx-rs (BENAMRAM, 2022), que realiza o processamento do dados em Rust na implementação do parser com uma interface python. Com esta biblioteca, em apenas alguns segundos a decodificação e leitura dos arquivos de *log* é realizada.

Optou-se também para realização das técnicas de *Curve Fitting* e testes de aderência sobre as distribuições candidatas o uso da linguagem R, por meio da biblioteca rpy2 (GAUTIER, 2022), que fornece uma interface Python para executar código R. Esta escolha se deu pela facilidade proveniente da linguagem R para realizar análise estatísticas de dados reais.

Na versão 2 do relatório, continuou-se utilizando a biblioteca rpy2 (GAUTIER, 2022) para execução dos códigos em R, e também foi utilizado a biblioteca Matplotlib (HUNTER, 2007), para a construção dos gráficos, uma boa solução para criar visualizações estáticas e interativas em Python.

5.2 Aprendizado

Realizar este trabalho de conclusão de curso me possibilitou obter diversos conhecimentos na área de confiabilidade de software, complementar o conteúdo que já foi visto em uma matéria da graduação é muito importante, pois coloca em prática o que foi visto e cria novos desafios ao aprofundar no assunto em questão.

Durante o desenvolvimento da versão 2 do X-RAT foi necessário também aprender sobre a forma que foi construída a primeira versão do sistema, que incluem, lidar com o processamento e operações em grandes quantidades de dados, editar e atualizar o banco de dados que armazenam as informações.

Foi necessário também aprender a lidar com a integração de programas construídos de formas diferentes, bem como realizar a implementação de algoritmos matemáticos e gráficos para melhor demonstração dos dados.

5.3 Dificuldades encontradas

Na parte inicial do projeto uma das dificuldades encontradas se deu pelo entendimento da forma que foi construída o software X-RAT, as suas funções e afins.

Outra dificuldade, e a principal delas, foi no aprimoramento do software, ao realizar a integração do algoritmo desenvolvido pelo Caio Santos (SANTOS; JR; TRIVEDI, 2020a) que realiza a previsão de falhas de software com base nos padrões de eventos múltiplos ao X-RAT, visto que é um assunto complexo a ser analisado, compreendido e integrado a plataforma.

Referências

AKAIKE, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, v. 19, n. 6, p. 716–723, 1974. Citado na página 25.

AMAZON. *pyevtx-rs*. 2023. Disponível em: <https://aws.amazon.com/pt/what-is/log-files/>. Acesso em: 15 junho 2023. Citado na página 11.

ANSI/IEEE. Standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, p. 1–84, 1990. Citado na página 10.

AVIZIENIS, A. et al. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, v. 1, p. 11–33, 2004. Citado na página 14.

BENAMRAM, O. *pyevtx-rs*. 2022. Disponível em: <https://github.com/omerbenamram/pyevtx-rs>. Acesso em: 11 junho 2023. Citado 2 vezes nas páginas 23 e 38.

BOTT, E.; SIECHERT, C.; STINSON, C. *Windows 10 Inside Out*. Microsoft Press, a division of Microsoft Corporation, 2016. (Inside out). ISBN 9781509304851. Disponível em: <https://books.google.com.br/books?id=8nbTjwEACAAJ>. Citado na página 12.

CULLEN, H. C. F. A. C. *Probabilistic techniques in exposure assessment : a handbook for dealing with variability and uncertainty in models and inputs / Alison C. Cullen and H. Christopher Frey*. New York: Plenum Press, 1999. ISBN 0306459566. Citado na página 25.

GANAPATHI, A.; PATTERSON, D. Crash data collection: a windows case study. In: *2005 International Conference on Dependable Systems and Networks (DSN'05)*. [S.l.: s.n.], 2005. p. 280–285. Citado na página 16.

GAUTIER, L. *pyevtx-rs*. 2022. Disponível em: <https://rpy2.github.io/>. Acesso em: 11 junho 2023. Citado 2 vezes nas páginas 25 e 38.

HIGH Performance and Dependable Computing Systems (HPDCS). 2011. Disponível em: <http://hpdcs.facom.ufu.br>. Acesso em: 27 maio 2021. Citado na página 17.

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007. Citado na página 38.

JR, R. M.; OLIVEIRA, G.; ARAUJO, L. Operating system reliability from the quality of experience viewpoint: An exploratory study. In: . [S.l.: s.n.], 2013. Citado 2 vezes nas páginas 24 e 25.

LYU, M. R.; NIKORA, A. P. Casre: a computer-aided software reliability estimation tool. *[1992] Proceedings of the Fifth International Workshop on Computer-Aided Software Engineering*, p. 264–275, 1992. Citado na página 14.

MICROSOFT. *pyevtx-rs*. 2012. Disponível em: <https://learn.microsoft.com/pt-br/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc722404(v=ws.10)>. Acesso em: 05 agosto 2023. Citado na página 12.

NETIKUS. *Event Sentry*. 2022. Disponível em: <https://www.eventsentry.com/>. Acesso em: 11 junho 2023. Citado na página 22.

OLIVEIRA, R. C. de. Windows 10: Uma análise de confiabilidade através de dados do usuário. 2018. Citado na página 17.

PEIXOTO, D. C. F. Geração automática de relatórios de confiabilidade de software: projeto e implementação de engine para a plataforma x-rat. 2023. Citado 10 vezes nas páginas 11, 12, 17, 20, 21, 22, 24, 26, 31 e 38.

RIBEIRO, G. F. *Profissionais TI*. 2012. Disponível em: <https://www.profissionaisti.com.br/alguns-dos-mais-famosos-erros-de-softwares-da-historia/>. Acesso em: 15 junho 2023. Citado na página 10.

RICARDO. *Empresas e Negócios*. 2022. Disponível em: <https://jornalempresasenegocios.com.br/tecnologia/boeing-737-max-erro-de-software-derruba-dois-avioes/>. Acesso em: 15 junho 2023. Citado na página 9.

ROBINSON, R. B. A. *ReportLab*. 2022. Disponível em: <https://www.reportlab.com/>. Acesso em: 11 junho 2023. Citado na página 26.

SANTOS, C.; JR, R. M. Failure patterns in operating systems: An exploratory and observational study. *Journal of Systems and Software*, v. 137, 04 2017. Citado na página 16.

SANTOS, C. A. dos; JR, R. M. Exploratory analysis on failure causes in a mass-market operating system. *ACM SIGOPS Operating Systems Review*, v. 50, p. 18–30, 03 2016. Citado na página 16.

SANTOS, C. A. dos; JR., R. M. Um estudo exploratório sobre padrões de falhas de sistemas operacionais. In: *Anais do XXX Concurso de Teses e Dissertações*. Porto Alegre, RS, Brasil: SBC, 2017. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/ctd/article/view/3456>. Citado 3 vezes nas páginas 16, 28 e 37.

SANTOS, C. A. dos; JR, R. M.; TRIVEDI, K. *A Statistical Approach to Predict Operating System Failure Based on Multiple Failures Association*. 2020. Citado 4 vezes nas páginas 17, 26, 27 e 39.

SANTOS, C. A. dos; JR, R. M.; TRIVEDI, K. *A Statistical Approach to Predict Operating System Failures Based on Multiple Failures Association*. 2020. Citado na página 12.

SANTOS, C. A. dos; JR, R. M.; TRIVEDI, K. *A Multisite Characterization Study on Failure Causes in System and Applications Software*. 2021. Citado na página 17.

SOFTWARE Dependability Laboratory (.SDLAB). 2020. Disponível em: <https://sdlab.facom.ufu.br/>. Acesso em: 15 junho 2023. Citado 2 vezes nas páginas 10 e 11.

STATCOUNTER Desktop Operating System Market Share. 2023. Disponível em: <https://gs.statcounter.com/os-market-share/desktop/worldwide>. Acesso em: 26 maio 2023. Citado na página 11.

TAWDE, S. *Critical System*. 2022. Disponível em: <https://www.educba.com/critical-system/>. Citado na página 9.

TEAM, T. pandas development. *pandas-dev/pandas: Pandas*. Zenodo, 2022. Disponível em: <https://doi.org/10.5281/zenodo.3509134>. Citado 2 vezes nas páginas 23 e 25.

THE Operating System Reliability Analysis Tool (OSRat). 2016. Disponível em: <http://hpdcs.facom.ufu.br/Software/osrat>. Acesso em: 27 maio 2021. Citado na página 17.

THE Operating System Reliability Analysis Tool (X-Rat). 2020. Disponível em: <http://hpdcs.facom.ufu.br/Software/osrat>. Acesso em: 15 junho 2023. Citado 2 vezes nas páginas 10 e 11.