UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Um Estudo Exploratório Sobre Padrões de Falhas de Software de Sistemas Operacionais

Caio Augusto Rodrigues dos Santos

Uberlândia, Minas Gerais
2016

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Um Estudo Exploratório Sobre Padrões de Falhas de Software de Sistemas Operacionais

Caio Augusto Rodrigues dos Santos

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Engenharia de *Software*
Orientador: Prof. Dr. Rivalino Matias Júnior

Uberlândia, Minas Gerais
2016

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

S237e 2016

Santos, Caio Augusto Rodrigues dos, 1989-
Um estudo exploratório sobre padrões de falhas de software de sistemas operacionais / Caio Augusto Rodrigues dos Santos. - 2016.
108 f. : il.

Orientador: Rivalino Matias Júnior.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Sistemas operacionais (Computadores) - Avaliação - Teses. 3. Falhas de sistemas de computação - Teses. 4. Software - Confiança - Teses. I. Matias Júnior, Rivalino. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "Um Estudo Exploratório Sobre Padrões de Falhas de Software de Sistemas Operacionais" por Caio Augusto Rodrigues dos Santos como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação.

Uberlândia, Março de 2016.

Orientador: _____

Prof. Dr. Rivalino Matias Júnior

Universidade Federal de Uberlândia

Banca Examinadora:

_____

Prof. Dr. Antônio Augusto Fröhlich

Universidade Federal de Santa Catarina

_____

Prof. Dr. Michel dos Santos Soares

Universidade Federal de Sergipe

_____

Prof. Dr. Paulo Romero Martins Maciel

Universidade Federal de Pernambuco

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: Março de 2016.

Autor:	**Caio Augusto Rodrigues dos Santos**
Título:	**Um Estudo Exploratório Sobre Padrões de Falhas de Software de Sistemas Operacionais**
Faculdade:	**Faculdade de Computação**
Grau:	**Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

_____

Autor

DEDICATÓRIA

Aos meus pais Adriano e Solange.
À minha irmã Natália.
Aos meus amigos.

AGRADECIMENTOS

Ao meu orientador, Prof. Rivalino Matias Júnior,
pela sua paciência, incentivo e dedicação.

"O caos é uma ordem por decifrar."

José Saramago

RESUMO

Estudos empíricos em confiabilidade de *software* têm predominantemente focado em aplicações de usuário. Devido à dependência intrínseca de programas de usuário em relação ao sistema operacional (SO), falhas no sistema operacional podem afetar, severamente, até mesmo as aplicações mais confiáveis. Portanto, entender como as falhas de SO ocorrem é um importante requisito para melhorar a confiabilidade de *software* como um todo. Esta pesquisa realizou um estudo exploratório sobre o comportamento das falhas de SO, com base em 7.007 registros de falha reais coletados de diferentes computadores executando um sistema operacional de mercado. Foi realizada uma análise quantitativa para investigar diferentes propriedades das falhas de SO analisadas. Os resultados indicam que os serviços de SO falharam mais do que qualquer outra categoria de falha de SO. Foi desenvolvido um protocolo de detecção de padrões de falhas, o qual permitiu detectar e caracterizar padrões de falhas que se mostraram consistentes em diferentes computadores dos mesmos e de variados ambientes de trabalho investigados. No total, foram detectados 45 padrões de falhas com 153.511 ocorrências. Com base nesses padrões, foram encontradas evidências empíricas de correlação entre falhas de componentes específicos do SO, uma propriedade importante para melhor compreender o comportamento dessas falhas de *software*. Além disso, o protocolo proposto foi adaptado para também detectar e caracterizar padrões específicos de causalidade entre as falhas de SO. Evidências empíricas confirmam a presença de correlação causal entre as falhas na amostra, em que tanto a correlação cruzada e a autocorrelação foram encontradas.

Palavras-chave: Sistemas operacionais; falhas de *software*; detecção de padrões; estudo empírico; causas de falha, correlação.

ABSTRACT

Empirical studies in software reliability have predominantly focused on end-user applications. Given the intrinsic dependency of user programs on the operating system (OS) software, OS failures can severely impact even the most reliable applications. Therefore, it is a major requirement to understand how OS failures occur in order to improve software reliability as a whole. This research carried out an exploratory study on OS failures behavior, based on 7,007 real failure records collected from different computers running a mass-market operating system. We performed a quantitative analysis to investigate different properties of the OS failures analyzed. The findings indicate that OS services failed more than any other OS failure category. We introduced an OS failure pattern detection protocol, which allow us to detect and characterize failure patterns that showed consistent across different computers from the same and varied workplaces investigated. In total, we detected 45 failure patterns with 153,511 occurrences. Based on these patterns, empirical evidences of correlation among specific OS components failures was found, which is a important property to understand better the behavior of these software failures. In addition, the proposed protocol was adapted to also detect and characterize specific causality patterns among the OS failures. Empirical evidences confirm the presence of causal correlation between the failures in the sample, where both cross-correlation and autocorrelation were found.

Keywords: Operating systems; software failures; pattern detection; empirical study; failure causes; correlation.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AIE	Atualização do Internet Explorer
AMO	Atualização do Microsoft Office
ANF	Atualização do .Net Framework
API	*Application Programming Interface*
AW7	Atualização do Windows
CBS	*Component Based Servicing*
Dll	*Dynamic-Link Library*
DP	Desvio Padrão
EFA	Evento de Falha Anterior
EFP	Evento de Falha Posterior
ERP	*Enterprise Resource Planning*
EXP	Explorer.exe
FR	Falha de Referência
GPU	*Graphics Processing Unit*
HW	*Hardware*
IDE	*Integrated Development Environment*
IE	Internet Explorer
IP	Intervalo de Pesquisa
KB	*Knowledge Base Identifier*
LEC	Lista de Eventos Candidatos
lPFC	Lista de PFC
lrpn	*List of Reference Previous Next*
MTBF	*Mean Time Between Failures*
N/A	*Not Applicable*
OSRat	*Operating System Reliability Analysis Tool*
PCI	*Peripheral Component Interconnect*
PF	Padrão de Falha
PFC	Padrão de Falha Candidato
RAC	*Reliability Analysis Component*
rf	Repositório de Falhas
RM	*Reliability Monitor*
SDF	SQL *Compact Edition Database File*
SO	Sistema Operacional

SO_{APP}	Aplicação de SO
SO_{KNL}	*Kernel* do SO
SO_{SVC}	Serviço de SO
SQL	*Structured Query Language*
SWU	Serviço *Windows Update*
TG	*TimeGenerated*
Win7	Microsoft Windows 7

1. INTRODUÇÃO

1.1 Contextualização

Sistemas computacionais se tornaram uma das mais importantes ferramentas da sociedade moderna. Devido ao aumento da dependência dos seres humanos por esses sistemas, visando principalmente produtividade, qualidade, segurança, conforto e praticidade, há também o aumento da preocupação com a confiabilidade destes sistemas.

Os sistemas computacionais podem ser empregados em diferentes práticas do nosso cotidiano, como comunicação, entretenimento e segurança. Em consequência dessa variedade de cenários em que estes sistemas são utilizados atualmente, uma falha em suas operações pode acarretar desde simples inconvenientes até grandes desastres [Matias *et al.* 2014]. Um exemplo disso é relatado em [Leveson e Turner 1993], em que uma máquina de radioterapia controlada por computador, chamada de Therac-25, causou uma série de acidentes entre 1985 e 1987, nos quais pacientes receberam overdoses de radiação, resultando em mortes e lesões graves. Ocasionalmente, a máquina aplicava doses até cem vezes maiores do que a normal em consequência de problemas no desenvolvimento do *software* responsável por controlar o feixe de energia.

A preocupação na confiança de sistemas computacionais não é exclusiva de sistemas críticos, especializados e customizados, mas também ocorre em sistemas ordinários. Por exemplo, o sistema de navegação utilizado pela marinha norte-americana executa sob um sistema operacional convencional, o Microsoft Windows [Li *et al.* 2008]. Isso indica que mesmo que o sistema computacional da marinha norte-americana seja especializado e com requisitos de alta confiabilidade, ele depende de um software de sistema operacional convencional, usado em computadores pessoais. Nesse caso, uma falha no sistema operacional Microsoft Windows pode impedir a execução de tarefas de alta criticidade no sistema de navegação. Cada vez mais sistemas computacionais críticos utilizam partes de software de propósito geral, em especial sistemas operacionais. Outro exemplo é o uso do sistema operacional Linux embarcado em diferentes sistemas computacionais [Bruzzone *et al.* 2006], [Abbott 2012].

De acordo com [Salfner *et al.* 2006], a maior parte das falhas em sistemas computacionais é causada pelo *software*. Segundo [Candea *et al.* 2003], a causa de inatividade nos sistemas computacionais mudou significativamente de *hardware* para *software*. Assim, surge maior necessidade de estudos voltados para a confiabilidade de *software*.

A confiabilidade de um *software* é definida como a probabilidade do *software* operar sem falhas durante um período de tempo específico em um dado ambiente [ANSI/IEEE 1991]. Entre outros atributos da qualidade de *software*, tais como funcionalidade (*functionality*), usabilidade (*usability*), capacidade (*capability*) e facilidade de manutenção (*maintainability*), a confiabilidade (*reliability*) é geralmente aceita como o principal fator de qualidade, uma vez que quantifica as falhas de *software*, as quais são as principais responsáveis pela inatividade de um sistema computacional [Lyu 2007].

De acordo com [Goseva-Popstojanova e Trivedi 2000a], diversos modelos analíticos têm sido propostos para abordar o problema da quantificação da confiabilidade de *software*. A maioria deles foca na estimação da confiabilidade do *software* por meio de seu histórico de falhas e, assim, tratam o *software* como uma caixa preta [Swift *et al.* 2003], assumindo algum modelo paramétrico do tempo entre as falhas ou do número de falhas ao longo de um intervalo de tempo finito. Por outro lado, os modelos que utilizam a abordagem caixa branca [Goseva-Popstojanova e Trivedi 2000b], estimam a confiabilidade por meio da representação da estrutura do *software* e são, em sua maioria, restritos à fase operacional do ciclo de vida do *software*.

Conforme apresentado em [Goseva-Popstojanova e Trivedi 2000a], [Goseva-Popstojanova e Trivedi 2000c] e [Goseva-Popstojanova e Trivedi 1999], uma das suposições realizadas pela maioria dos modelos de confiabilidade de *software* é tratar as falhas como eventos independentes. Em [Nikora e Lyu 1996] é enfatizado que as suposições de independência entre as falhas são realizadas para adequar os modelos para formas matematicamente tratáveis e simplificar a estimativa dos parâmetros dos modelos. Portanto, essas suposições podem não representar a real ocorrência das falhas e, assim, podem prejudicar a estimação da confiabilidade do *software*.

Para evitar suposições que podem não representar o real comportamento das falhas durante a estimação da confiabilidade de um *software*, é essencial compreender a maneira com que as falhas ocorrem em condições reais de funcionamento desse *software*. Portanto, a análise de dados de falha coletados durante a fase de testes do *software* pode ser limitada, e talvez não represente todos os cenários que ocorreriam em um ambiente operacional real.

Ressalta-se que a maioria das aplicações em nível de usuário deve funcionar sob um sistema operacional (SO). Dessa forma, o sistema operacional deve apresentar um nível de confiabilidade que garanta a confiança das aplicações em nível de usuário nos serviços prestados por ele. Entretanto, os sistemas operacionais modernos enfrentam um grande desafio para garantir tal confiabilidade, pois devem suportar diferentes tipos de carga de trabalho (*workload*) de usuários, *device drivers* e acompanhar inovações de *hardware*, causando um grande aumento no seu tamanho e complexidade [Murphy 2008]. De acordo com a literatura, o aumento no tamanho do *software* (ex. linhas de código) favorece a ocorrência de falhas [Lipow 1982] e [Murphy 2004]. De igual forma, a ocorrência de falhas é afetada pelo grau de complexidade do software, tal como preconizado por Dijkstra: "*Simplicity is prerequisite for reliability*" [Dijkstra 1975].

Este trabalho de pesquisa visou aprimorar a compreensão do comportamento das ocorrências de falhas de sistemas operacionais, investigando tanto a correlação entre elas quanto os eventos que as antecedem e sucedem, com o propósito de detectar e caracterizar padrões que possam ser usados para auxiliar em estimativas mais precisas da confiabilidade dos sistemas operacionais. Além disso, foram analisadas as possíveis causas dessas falhas, e como a causa de uma falha está relacionada com a causa de outras falhas.

1.2 Relevância da Pesquisa

A confiabilidade de *software* impacta diretamente no funcionamento de sistemas computacionais. Nesse sentido, garantir a confiabilidade é um importante requisito para esses sistemas. Uma revisão da literatura de confiabilidade de *software* mostrou que a maioria dos estudos nesta área têm-se concentrado em confiabilidade das aplicações de usuário. Porém, como mencionado anteriormente, a maioria dessas aplicações computacionais depende de um SO. Portanto, o SO deve garantir, pelo menos, o mesmo nível de confiabilidade esperado para as aplicações de usuário para que a confiabilidade de todo o sistema não seja comprometida [Matias *et al.* 2014].

Em [Antunes e Matias 2014], os autores consideraram que mesmo que o *hardware* e as aplicações em nível de usuário apresentem alta confiabilidade, a confiabilidade do sistema como um todo é prejudicada se o SO não apresentar a confiabilidade necessária. Isso pode ser observado quantitativamente no exemplo fictício da Figura 1.1, em que apesar das aplicações em nível de usuário e do *hardware* possuírem alta confiabilidade (99,999%), a confiabilidade do sistema computacional é diminuída pela confiabilidade do SO.

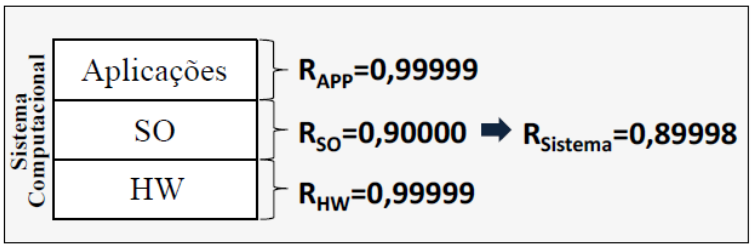


Figura 1.1. Confiabilidade sistêmica.
Fonte: [Antunes e Matias 2014]

Alguns trabalhos (ex. [Kalyanakrishnam *et al.* 1999], [Xu *et al.* 1999], [Chou *et al.* 2001], [Swift *et al.* 2003], [Ganapathi and Patterson 2005], [Ganapathi *et al.* 2006] e [Murphy 2008]) investigaram a confiabilidade de sistemas operacionais. No entanto, todos eles consideraram apenas as falhas que ocorreram em nível de *Kernel* do SO como falhas de sistema operacional. Em [Matias *et al.* 2013] ressalta-se que sistemas operacionais modernos possuem partes executando tanto em nível de *Kernel* quanto de usuário, portanto, as pesquisas que focam suas análises em falhas que ocorreram apenas no nível de *Kernel* do SO não conseguem avaliar a real confiabilidade do SO. Por esta razão, o estudo apresentado nesta dissertação considerou falhas relacionadas com os dois níveis de execução do sistema operacional, isto é, em nível de *Kernel* e de usuário.

Por fim, nenhum trabalho que aborda a temática de detecção e caracterização de padrões e/ou correlação de falhas de sistemas operacionais foi encontrado na literatura. Por meio da investigação dessa temática é possível melhorar a compreensão do comportamento dessas falhas, o que pode ser utilizado, posteriormente, na prevenção de falhas, e assim, aumentar a confiabilidade dos sistemas computacionais como um todo.

1.3 Objetivos da Pesquisa

Geral:

- Investigar a existência de padrões de falhas de sistema operacional. Nesta pesquisa, padrões de falhas são combinações de eventos de falhas de SO que se repetem sistematicamente em diversos computadores nos diferentes ambientes de trabalho analisados.

Específicos:

- Definir uma abordagem sistematizada (protocolo) para detectar e caracterizar padrões de falhas de SO, aplicável a dados de falha de qualquer SO.
- Testar a hipótese de dependência entre as falhas de SO.
- Detectar e caracterizar correlações entre falhas de SO.
- Detectar e caracterizar correlações entre as causas das falhas de SO.

Como não foram encontrados trabalhos anteriores relacionados com a temática de padrões de falhas de SO, foi necessário definir uma abordagem própria, na forma de um protocolo, para detectar e caracterizar esses padrões.

A hipótese de dependência entre as falhas de SO foi testada por ser uma propriedade que é muitas vezes negligenciada pelos estudos nesta área. Este teste foi realizado a partir de uma análise exploratória dos dados de falha analisados.

A correlação entre as falhas de SO foi analisada com o propósito de identificar a relação entre as ocorrências dessas falhas, visando identificar possíveis ocorrências de correlações cruzadas e autocorrelações.

Já na correlação entre as causas das falhas de SO, o objetivo foi investigar a relação entre as causas das falhas observadas na amostra.

1.4 Desenvolvimento da Pesquisa

Nesta seção estão descritas, de modo geral, as principais atividades que foram realizadas nesta pesquisa de mestrado. Esta pesquisa realizou um estudo de natureza exploratória. De acordo com [Shields e Rangarjan 2013], esse tipo de estudo é realizado para problemas que não foram claramente compreendidos e descritos na literatura disponível. Portanto, o estudo exploratório pode ser executado como a primeira análise de um determinado problema. Dessa forma,

esse tipo de estudo auxilia na determinação das melhores abordagens utilizadas para tratá-lo posteriormente.

No contexto do estudo apresentado nesta dissertação, o problema analisado são os padrões de falhas de SO. Primeiramente, na ausência de trabalhos similares, este estudo desenvolveu uma abordagem própria para verificar a existência desse fenômeno em ambientes reais de trabalho e depois caracterizá-lo. Portanto, os resultados obtidos neste estudo representam o comportamento das falhas de SO como observado em ambientes reais de trabalho.

1.4.1 Revisão da Literatura

Nesta etapa foram estudados tópicos relacionados com o tema "confiabilidade de *software*". Posteriormente, foi realizada uma revisão da literatura com foco em estudos relacionados com falhas de *software* de sistema operacional, em especial, trabalhos voltados para a investigação de padrões e/ou correlação dessas falhas.

A busca visou localizar qualquer trabalho relacionado com os temas mencionados anteriormente, sem restringir a data de publicação. As principais fontes de pesquisa nessa etapa foram livros, publicações em revistas e anais de conferências científicas especializadas. Essas fontes foram obtidas por meio de bibliotecas digitais, tais como IEEE *Xplore Digital Library* e ACM *Digital Library,* e utilizando ferramentas de pesquisa como o *Google Scholar*.

1.4.2 Material

Dado que essa pesquisa focou na compreensão do comportamento das ocorrências de falha de sistemas operacionais, as amostras de falha de SO utilizadas neste trabalho foram obtidas de sistemas reais em execução.

A amostra de trabalho é composta de 7.007 registros de falha de SO obtidos de 566 computadores executando o sistema operacional Microsoft Windows 7 (Win7). Esses registros foram obtidos de quatro ambientes reais de trabalho com diferentes perfis de utilização.

Os registros de falha analisados são provenientes de diferentes ambientes de uso, em especial esses dados foram coletados de computadores instalados em ambientes acadêmicos, corporativo e de computadores pessoais/domésticos.

Com o objetivo de simplificar a identificação dessas amostras, as mesmas foram organizadas em quatro grupos, nomeados como G1, G2, G3 e G4. Uma descrição detalhada da amostra de trabalho é encontrada no Capítulo 3.

1.4.3 Métodos

Este trabalho realizou um estudo exploratório sobre a confiabilidade de um SO de propósito geral. Embora os resultados deste trabalho sejam específicos ao SO investigado, os métodos utilizados podem ser aplicados a qualquer SO de interesse.

Primeiramente foi proposta uma abordagem para agrupar e classificar diferentes tipos de falha de SO. Posteriormente, um protocolo de detecção e caracterização de padrões de falhas de SO foi proposto, permitindo a detecção de padrões de falhas considerados consistentes em diferentes computadores de variados ambientes de trabalho.

Por fim, o protocolo proposto foi adaptado também para detectar e caracterizar as relações causais de falhas de SO. Foram realizadas análises estatísticas quantitativas para investigar as propriedades das falhas de SO encontradas, bem como suas causas e correlações.

1.5 Estrutura do Documento

Os demais capítulos desta dissertação estão organizados da seguinte forma.

O Capítulo 2 trata da fundamentação teórica que serviu de base para este estudo, apresentando conceitos teóricos de falhas de *software*, o tipo de categorização realizado nas falhas de sistemas operacionais e a revisão da literatura correlata.

O Capítulo 3 faz a descrição da amostra de dados de falha de SO. Primeiramente, o sistema operacional investigado é identificado. Posteriormente, é apresentado o mecanismo de coleta das falhas, os agrupamentos realizados e a descrição da análise estatística dos dados. Por fim, é apresentada a descrição detalhada da amostra de trabalho utilizada.

O Capítulo 4 apresenta o protocolo criado para detectar e categorizar padrões de falhas de SO, juntamente com uma taxonomia criada para definir vários conceitos utilizados no protocolo. Por fim, foi apresentada a adaptação do protocolo para detectar e caracterizar padrões de causalidade entre as falhas de SO.

O Capítulo 5 apresenta os resultados do estudo exploratório. Nesse capítulo, o protocolo criado é aplicado à amostra de dados de falha descrita no Capítulo 3, apresentando a análise dos resultados e os principais achados obtidos.

Por fim, o Capítulo 6 apresenta as considerações finais sobre a pesquisa, destacando os principais resultados e conclusões, as limitações da pesquisa, as contribuições para a literatura, as dificuldades encontradas e os trabalhos futuros.

2. REVISÃO DA LITERATURA

2.1 Introdução

A taxonomia de falhas de *software* adotada neste trabalho foi definida em [Avižienis *et al.* 2004]. Este artigo apresenta as principais definições relacionadas com dependabilidade (confiança no funcionamento) de sistemas computacionais. A dependabilidade é um dos aspectos da qualidade de um sistema. Esta área se preocupa com a confiança no funcionamento dos sistemas tanto de *software* quanto de *hardware*. Portanto, a dependabilidade de um sistema computacional é a habilidade deste sistema em fornecer um serviço no qual seu usuário pode, justificadamente, ter confiança no seu funcionamento. Os atributos da dependabilidade definidos em [Avižienis *et al.* 2004] são:

- **Disponibilidade** (*availability*): prontidão (*readiness*) para o serviço correto.
- **Confiabilidade** (*reliability*): continuidade do serviço correto.
- **Segurança** (*safety*): ausência de consequências catastróficas sobre o usuário e ambiente.
- **Integridade** (*integrity*): ausência de alterações impróprias no sistema.
- **Manutenibilidade** (*maintainability*): habilidade de sofrer modificações e reparos.

Considerando esses atributos da dependabilidade, esta pesquisa concentra-se na confiabilidade, especificamente, na confiabilidade do *software* de sistema operacional. Os elementos que permitem calcular a confiabilidade de um *software* são, principalmente, suas falhas. De acordo com [Lewis 1995], o tempo de ocorrência das falhas é a variável mais importante quando se trata de confiabilidade. Tanto os modelos analíticos de caixa preta (ex. [Swift *et al.* 2003], [Ganapathi e Patterson 2005] e [Ganapathi *et al.* 2006]), que estimam a confiabilidade do *software* por meio de seu histórico de falhas, quanto os modelos que utilizam a abordagem caixa branca (ex. [Goseva-Popstojanova e Trivedi 2000c] e [Dai *et al.* 2005]), estimando a confiabilidade por meio da representação da estrutura do *software*, necessitam de informações de caracterização do comportamento das ocorrências das falhas. Portanto, a análise das ocorrências das falhas de um *software* é essencial para permitir que o cálculo da sua confiabilidade represente integralmente a real percepção de sua confiabilidade.

A Seção 2.2 explica conceitos sobre falhas de *software* e apresenta uma descrição teórica de como essas falhas ocorrem de forma geral. Além disso,

descreve como as falhas de sistemas operacionais foram consideradas neste trabalho.

Por fim, a Seção 2.3 apresenta trabalhos encontrados na literatura que abordam assuntos relacionados com o tema confiabilidade de *software*. Dentre eles, detacam-se trabalhos relacionados com análise quantitativa de confiabilidade de sistemas operacionais e desenvolvimento de modelos para estimar a confiabilidade de *software* levando em consideração a correlação entre as falhas.

2.2 Falhas de Software

Falta, erro e falha de *software* são as três ameaças à dependabilidade computacional [Avižienis *et al.* 2004]. Alguns conceitos definidos em [Avižienis *et al.* 2004] estão descritos a seguir para auxiliar no entendimento de cada uma dessas ameaças.

- **Sistema:** entidade que interage com outras entidades. Isto é, outros sistemas, incluindo *hardware*, *software*, ser humano.
- **Ambiente:** outros sistemas que interagem com um determinado sistema são denominados como o ambiente deste sistema.
- **Fronteira de um sistema:** é a fronteira comum entre um sistema e seu ambiente.
- **Função de um sistema:** o que o sistema se destina a fazer.
- **Comportamento de um sistema:** o que o sistema faz para implementar a sua função.
- **Usuário**: é outro sistema que recebe o serviço do sistema provedor.
- **Serviço prestado:** o serviço prestado por um sistema provedor é o seu comportamento como é percebido por um sistema usuário.
- **Interface de serviço:** parte da fronteira do sistema do provedor em que a prestação de serviços ocorre.
- **Estado interno e externo:** estado externo é a parte do estado total do provedor que é perceptível na interface de serviço; a parte restante é o seu estado interno. O serviço prestado é uma sequência de estados externos do provedor.
- **Interface de uso:** interface do sistema usuário que recebe o serviço fornecido pelo sistema provedor.

O **serviço** prestado por um **sistema** é classificado como correto quando o **serviço entregue** ao seu usuário condiz com a **função do sistema**, ou seja, o serviço prestado está de acordo com o que o sistema se destina a fazer. A **falha** ocorre quando o serviço prestado não está em conformidade com a especificação funcional, ou quando esta especificação não descreve adequadamente a função do sistema. Considera-se que um sistema falhou somente quando sua saída é recebida pelo usuário do sistema e é percebida como um resultado inesperado ou errado, ou seja, o serviço prestado não segue a especificação funcional determinada.

Como o serviço é uma sequência de **estados externos** de um sistema, uma falha no serviço significa que, pelo menos, um (ou mais) estado(s) externo(s) do sistema desviou do estado de serviço correto. Esse desvio é chamado de **erro**.

Portanto, um **erro** é a parte do **estado interno** do sistema que pode levar à falha do serviço. Erros podem ser transformados em outros erros, esta transformação é chamada de propagação de erro. A propagação de um erro leva um sistema a falhar se o erro é propagado até a interface de serviço do sistema, provocando o desvio do serviço em relação a sua especificação, sendo percebido pelo usuário do sistema como um resultado diferente do esperado. Nota-se que apesar da percepção do erro ocorrer no momento da entrega do serviço, o sistema já poderia estar com erros muito tempo antes. Portanto, uma falha é a percepção, realizada pelo usuário do sistema, de um erro no serviço fornecido. Muitos erros não atingem o estado externo do sistema e, portanto, não causam uma falha.

A causa de um erro é a ativação de uma **falta** (defeito ou *bug*). Uma falta é considerada dormente até sua ativação. A ativação de uma falta causa um erro, o qual pode ou não alcançar a interface de serviço do sistema.

As três ameaças à dependabilidade possuem uma relação causal denominada de cadeia fundamental da dependabilidade. Esta cadeia é apresentada na Figura 2.1.

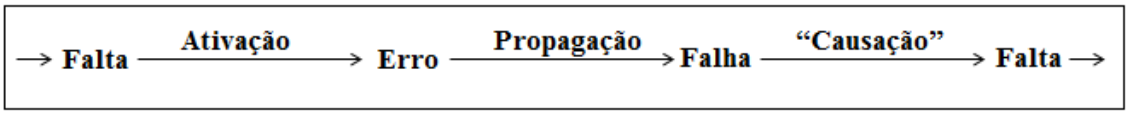


Figura 2.1. Cadeia fundamental da dependabilidade.
Fonte: [Antunes e Matias 2014]

Quando uma falta, anteriormente dormente, é ativada por um processo computacional, ela é chamada de falta interna. Neste caso a execução de um determinado *input* por um componente causa a ativação da falta dormente. Por exemplo, a tipificação incorreta de uma variável na linguagem de programação C, em que uma variável é declarada como *int* no lugar de *long int* levará a um erro somente quando a variável assumir valores do tipo *long int*, os quais irão ultrapassar a capacidade de armazenamento do tipo *int* causando um *overflow*. A ativação de uma falta externa ocorre quando a falha de um sistema provoca uma falta (externa) em outro sistema (usuário) que recebe os serviços do sistema provedor que falhou. Este caso é observado no final da representação da cadeia fundamental da dependabilidade (Figura 2.1).

Para [Avižienis *et al.* 2004], existem classes elementares de faltas classificadas de acordo com oito pontos de vista, as quais são:

- Fase de criação ou ocorrência

  Faltas de desenvolvimento: ocorrem durante (a) desenvolvimento do sistema, (b) manutenção durante a fase de uso e (c) geração de procedimentos para operar ou manter o sistema.

  Faltas de operação: ocorre durante a entrega do serviço na fase de uso.

- Fronteiras do sistema

Faltas internas: originam-se dentro da fronteira do sistema.

Faltas externas: originam-se fora da fronteira do sistema e propagam erros dentro do sistema por meio de interações e interferências.

- Causa fenomenológica

  Faltas naturais: causadas por fenômenos naturais sem a participação humana.

  Faltas humanas: resultado de ações humanas.

- Dimensão

  Faltas de *hardware*: originam-se ou afetam o *hardware*.

  Faltas de *software*: afetam o *software*, ou seja, programas e dados.

- Objetivo

  Faltas maliciosas: introduzidas por um ser humano com objetivos maliciosos de prejudicar o sistema.

  Faltas não maliciosas: introduzidas sem um objetivo malicioso.

- Intenção

  Faltas deliberadas: resultado de uma decisão prejudicial.

  Faltas não deliberadas: introduzidas sem intenção.

- Capacidade

  Faltas acidentais: introduzidas inadvertidamente.

  Faltas de incompetência: resultado da falta de competência profissional de uma pessoa responsável ou pela inadequação da organização de desenvolvimento.

- Persistência

  Faltas permanentes: assume-se que sua presença é contínua com o tempo.

  Faltas transientes: sua presença é ligada ao tempo.

Diferentes causas raiz das falhas (faltas) foram apresentadas de acordo com diferentes pontos de vista. As falhas também possuem diferentes formas de ocorrência. Essencialmente, uma falha é um evento que ocorre quando o serviço prestado por um sistema desvia-se de sua especificação funcional e é percebido pelo usuário do sistema como um serviço diferente do esperado. Modos de falha do serviço de um sistema é a denominação dada para as diferentes formas em que seu desvio é manifestado. De acordo com [Avižienis *et al.* 2004] existem quatro modos de falha do serviço de um sistema. Esses modos estão descritos a seguir.

- **Falhas de conteúdo** (*Content failures*)**:** o conteúdo das informações entregue na interface de serviço desvia da implementação da função do sistema.
- **Falhas de temporização** (*Timing failures*)**:** o tempo de chegada ou da duração das informações apresentadas na interface de serviço (ou seja, no

momento da prestação dos serviços) se desvia da implementação da função do sistema.

- **Falhas de interrupção** (*Halt failures*)**:** quando o serviço é interrompido; um caso especial de interrupção é a falha silenciosa, que ocorre quando nenhum serviço é entregue na interface de serviço.
- **Falhas erráticas** (*Erratic failures*)**:** quando o serviço é prestado (não é interrompido), mas é irregular.

Por fim, os autores de [Avižienis *et al.* 2004] indicam que a presença de dois ou mais usuários em um sistema possibilita a distinção da consistência das falhas ocorridas nesse sistema. Falhas consistentes ocorrem quando o mesmo serviço incorreto é percebido por todos os usuários. Por outro lado, quando diferentes serviços incorretos são percebidos por alguns ou todos os usuários do sistema, têm-se as chamadas falhas inconsistentes (alguns usuários podem até perceber o serviço correto).

A seguir tem-se a descrição de como as falhas de sistemas operacionais foram consideradas neste trabalho.

2.2.1 Falhas de Sistemas Operacionais

Alguns trabalhos anteriores (ex. [Swift *et al.* 2003], [Ganapathi e Patterson 2005] e [Ganapathi *et al.* 2006]) investigaram a confiabilidade de sistemas operacionais por meio de suas falhas. No entanto, todos estes trabalhos consideram como falhas de SO apenas as falhas que ocorrem em nível de *Kernel*. De acordo com [Matias *et al.* 2014], não importa se os componentes do sistema operacional falham em nível de usuário ou nível de *Kernel*, em ambos os casos a experiência do usuário é afetada.

Um exemplo apresentado pelos autores de [Matias *et al.* 2014] sobre componentes do sistema operacional falhando em nível de usuário é a falha da aplicação de SO de gerenciamento de janelas, existente em sistemas operacionais que possuem interface gráfica, como o programa explorer.exe presente no sistema operacional Win7. De acordo com os autores, a interrupção do funcionamento dessa aplicação pode afetar a experiência do usuário, impedindo-o de acessar arquivos e de executar programas. Neste caso, mesmo com o *Kernel* do SO executando corretamente, do ponto de vista do usuário o SO falhou. Com base em análises quantitativas, os autores demonstram que métricas de confiabilidade que consideram todos os níveis são mais representativas da real percepção dos usuários em relação à confiabilidade do sistema operacional do que métricas que consideram apenas o nível do *Kernel*.

Em [Matias *et al.* 2013], foi apresentada uma abordagem que analisou as falhas de sistema operacional em dois níveis (espaço do usuário e *Kernel*). Todas as falhas de SO coletadas foram classificadas em três categorias: Aplicação de SO (SO_{APP}), Serviço de SO (SO_{SVC}) e *Kernel* (SO_{KNL}). As duas primeiras categorias contêm os registros de falha em programas que executam em nível de usuário, enquanto a última de subsistemas de SO que executam em nível de *Kernel*.

A primeira categoria contém as falhas causadas pelo funcionamento incorreto das aplicações de SO. Essas aplicações geralmente executam em primeiro plano (*foreground*) e realizam funções específicas que são requisitadas de acordo com a necessidade do usuário do sistema. Um exemplo de aplicação de SO é o programa explorer.exe, no sistema operacional Win7. Esse programa é o responsável pelo gerenciamento de janelas e da manutenção/acesso aos arquivos neste SO.

A segunda categoria contém as falhas de serviços de SO. A maioria desses serviços é iniciada durante a carga do SO e executa em segundo plano (*background*). Normalmente, os serviços de SO são utilizados por outros componentes de SO. Portanto, suas funcionalidades não estão diretamente ligadas à necessidade do usuário do sistema. Como exemplo pode-se citar o svchost.exe, que possui a função de hospedar outros serviços individuais do Win7 que são implementados por *Dynamic-link Libraries* (Dlls) [Anson *et al.* 2012]. O svchost.exe é a interface entre as Dlls de serviços do Win7 e os demais componentes do sistema. Existem várias instâncias do svchost.exe em execução no Win7 e cada instância acomoda diferentes serviços. A Figura 2.2 apresenta um exemplo extraído de [Anson *et al.* 2012], em que o svchost.exe hospeda dois serviços: Dhcp e Dnscache. Quando o sistema operacional executa esses serviços, uma instância do svchost é iniciada para os dois serviços. Dessa forma, a instância svchost carrega as Dlls necessárias para executar cada serviço.

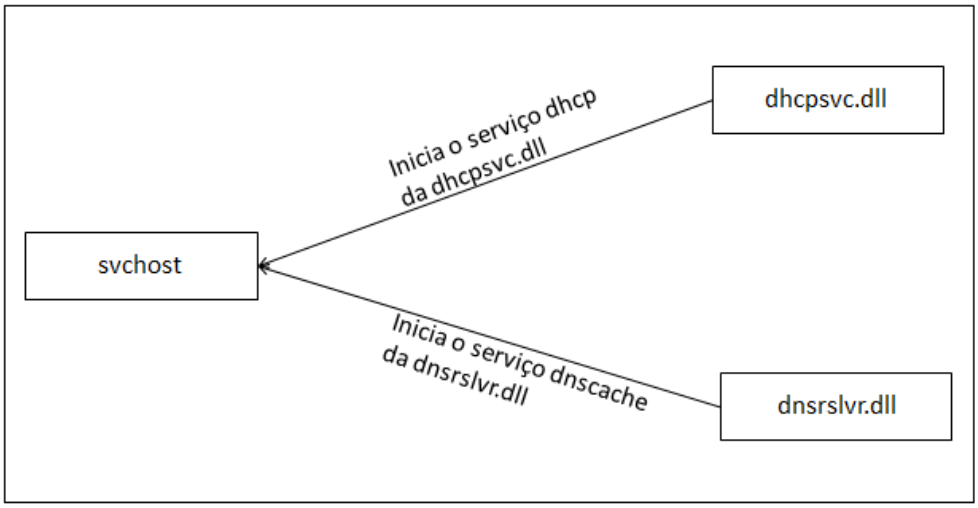


Figura 2.2. Instância do svchost carregando as Dlls necessárias para executar os serviços que elas implementam.
Fonte: Traduzido de [Anson *et al.* 2012]

Por fim, a terceira categoria possui falhas causadas pelos subsistemas do *Kernel* do SO, as quais ocorrem no espaço do *Kernel* e, geralmente, exigem a reinicialização do sistema. As falhas nesta categoria estão fortemente relacionadas às bibliotecas do sistema (ex. ntdll.dll e *Kernel*base.dll) e *device drivers*. Um exemplo de falha nesta categoria é a VIDEO_TDR_ERROR. Esta falha ocorre quando o controlador do dispositivo de vídeo detecta que já não pode controlar a unidade de processamento gráfico. Isso indica uma falha durante a tentativa de recarregar o *driver* de vídeo [Russinovich *et al.* 2009].

Analisando os resultados obtidos em [Matias *et al.* 2014], observou-se que após a categorização de 30.000 registros de falha reais do Win7, provenientes de diferentes ambientes de trabalho, a maior parte das falhas de SO ocorreram em nível de usuário. Portanto, estudos de confiabilidade de sistemas operacionais não devem analisar apenas as falhas que ocorrem em nível de *Kernel*, tal como tem sido a norma na literatura.

O estudo apresentado nesta dissertação utilizou a abordagem originalmente proposta em [Matias *et al.* 2013], a qual considera as falhas de sistemas operacionais nos dois níveis (usuário e *Kernel*), classificando essas falhas em três categorias (Aplicação de SO (SO_{APP}), Serviço de SO (SO_{SVC}) e *Kernel* (SO_{KNL})). Uma descrição detalhada de como as falhas de SO analisadas foram classificadas é apresentada no Capítulo 3.

2.3 Trabalhos Correlatos

Após a realização de uma revisão da literatura, não foram encontrados trabalhos que tratam especificamente de detecção e caracterização de padrões e/ou correlação de falhas de *software* de sistemas operacionais. Os trabalhos encontrados que mais se relacionam com esse tema investigaram a confiabilidade de sistemas operacionais por meio de seus históricos de falha; outros investigaram os tipos de falha mais recorrentes em um sistema computacional e alguns trabalhos propuseram modelos para estimar a confiabilidade de *software* levando em consideração a correlação entre as falhas.

A partir de 1.546 falhas coletadas de mais de 200 computadores de um ambiente acadêmico utilizando o sistema operacional Microsoft Windows XP SP1, os autores de [Ganapathi e Patterson 2005] concluíram que falhas de aplicação são mais frequentes do que de *Kernel*, porém, as falhas de *Kernel* geralmente exigem a reinicialização completa do computador.

Em [Ganapathi *et al.* 2006], os autores analisaram 2.528 ocorrências de falhas de *Kernel* do Windows XP, coletadas de 617 computadores. Concluiu-se que a maioria das falhas foi causada por códigos de *device drivers* de terceiros. Portanto, o sistema operacional Windows não foi o responsável por grande parte das falhas observadas.

Assim como as conclusões obtidas em [Ganapathi *et al.* 2006], os autores de [Swift *et al.* 2003] relataram que falhas de sistemas operacionais são principalmente causadas por extensões de *Kernel* (*device drivers*). Os autores destacaram que, à época, os *device drivers* eram responsáveis por mais de 70% do código do *Kernel* Linux e o Windows XP apresentou mais de 35.000 *device drivers* diferentes. Eles também concluíram que as extensões de terceiros, do *Kernel* de SO, foram uma das principais causas de falha do Windows XP, sendo que os *device drivers* foram responsáveis por 85% das falhas relatadas.

Nota-se que os trabalhos citados anteriormente consideraram apenas as falhas em nível do *Kernel* como falhas de sistema operacional. Como já discutido na Seção 2.2.1, de acordo com [Matias *et al.* 2013], esta abordagem não reflete a real experiência dos usuários com relação às suas percepções das falhas de SO, pois os sistemas operacionais possuem partes executando em nível de usuário e falhas

nessas partes também impactam as funcionalidades implementadas pelo sistema operacional, portanto, devem ser consideradas.

Em [Matias *et al.* 2014], os autores utilizaram a abordagem prosposta em [Matias *et al.* 2013], a qual considera não apenas as falhas de *Kernel* como falhas de sistema operacional, mas também as falhas de aplicação e serviço de SO. Os autores concluíram que as falhas de serviços de SO prevaleceram em relação às falhas de aplicações de SO e *Kernel*, as quais tiveram quantidades similares. Com base na análise estatística dos tempos das falhas, isto é, o registro realizado pelo SO do horário que uma falha ocorreu, os autores descobriram que a Gama e a Weibull foram as distribuições de probabilidades que melhor se ajustaram aos tempos de falhas, o que permitiu o cálculo das curvas de confiabilidade por grupos e categorias de falhas.

Como destacado por [Goseva-Popstojanova e Trivedi 1999], [Goseva-Popstojanova e Trivedi 2000a] e [Goseva-Popstojanova e Trivedi 2000c], a maioria dos estudos de confiabilidade de *software* trata as falhas de *software* como eventos independentes. Em alguns casos, essa suposição é realizada para simplificar a análise da confiabilidade [Nikora e Lyu 1996]. Em outros casos, essa suposição ocorre devido à falta de evidências empíricas com respeito à dependência entre as falhas de interesse. Em ambos os casos, a confiabilidade estimada por esses modelos pode diferir, significativamente, da real confiabilidade do *software* analisado, já que ao considerarem as ocorrências das falhas de forma independente, o cálculo de confiabilidade despreza a importante influência dos efeitos da correlação.

Em [Goseva-Popstojanova e Trivedi 2000c], foi desenvolvido um *framework* de modelagem de confiabilidade de *software* que contempla o fenômeno da correlação entre as falhas e, assim, estuda seus efeitos nas medidas de confiabilidade. No entanto, de acordo com os próprios autores, para a aplicação do *framework* em cenários reais, é necessário o desenvolvimento de modelos mais detalhados e específicos dentro do *framework*, assim como o desenvolvimento de inferências estatísticas para os parâmetros dos modelos. Tais inferências somente são possíveis com base em dados de falha reais, etapa que não foi realizada pelos autores.

Estendendo os resultados obtidos em [Goseva-Popstojanova e Trivedi 2000c], os autores de [Dai *et al.* 2005] desenvolveram um modelo de confiabilidade de *software* com base em Markov *Renewal Process*, o qual considera a correlação entre as falhas, além de incluir os casos de reinicialização do sistema com e sem reparo depois de uma falha de *software*. Os autores explicam que embora seu modelo apresente maior complexidade do que a abordagem tradicional, baseada no crescimento da confiabilidade, ele incorpora mais informações sobre as falhas e a estrutura do sistema.

Em [Goseva-Popstojanova e Trivedi 2000c] e [Dai *et al.* 2005] foram propostos modelos teóricos para estimar a confiabilidade de *software* com base na correlação entre as falhas, porém, o comportamento das falhas em um ambiente real não foi analisado. Em um trabalho recente, [Bird *et al.* 2014], seus autores demonstraram que a avaliação da confiabilidade de um *software* em um único contexto pode produzir uma estimativa incompleta e imprecisa de sua

confiabilidade no mundo real, por conta da influência do ambiente do sistema em que estiver inserido, do qual o *hardware* e a carga de trabalho (*workload*) são fatores de alto impacto.

Em [Bird *et al.* 2014], dados de falha de mais de 200.000 computadores executando sistemas operacionais Microsoft Windows foram analisados. Foi investigada a influência ambiental tanto de *hardware* quanto de software em falhas de aplicações de usuário. Apesar de o estudo focar em falhas de aplicações de usuário, suas conclusões corroboram os principais resultados obtidos nesta dissertação, pois mostram que a confiabilidade de aplicações individuais está fortemente relacionada com outras aplicações instaladas no computador. Por exemplo, eles concluíram que jogos e aplicações de compartilhamento de arquivos tendem a diminuir a confiabilidade de outras aplicações, enquanto aplicações de segurança tendem a aumentá-la. Especialmente em termos de correlação das falhas, os autores consideraram o cenário em que a atualização de uma aplicação pode afetar outras aplicações. Por exemplo, eles discutem o caso em que duas aplicações compartilham a mesma biblioteca e, ao atualizar uma das aplicações, essa ação altera a biblioteca compartilhada, o que pode provocar uma falha na outra aplicação. Situações semelhantes foram observadas no estudo apresentado nesta dissertação, em que bibliotecas compartilhadas (Dlls) causaram certos tipos de falha.

Como visto em [Swift *et al.* 2003], [Ganapathi e Patterson 2005], [Ganapathi *et al.* 2006], e [Matias *et al.* 2014], geralmente estudos empíricos em confiabilidade de *software* de sistemas operacionais não investigam a correlação entre as falhas de SO. O estudo apresentado nesta dissertação não apenas apresenta evidências empíricas de dependência entre falhas de sistema operacional, mas também investiga a natureza dessa dependência, ou seja, a forma como as diferentes falhas de sistema operacional se relacionam entre si.

Diferentemente de [Goseva-Popstojanova e Trivedi 2000c] e [Dai *et al.* 2005] que criaram modelos teóricos para estimar a confiabilidade de *software* com base na correlação entre as falhas, porém, sem utilizar informações sobre dados reais de falhas, o presente estudo leva em consideração a correlação entre falhas de SO, investigando justamente o comportamento das ocorrências dessas falhas em ambientes reais de trabalho.

3. AMOSTRA DE DADOS

3.1 Introdução

O objetivo deste capítulo é descrever a amostra de dados de falha de SO usada nesse estudo, bem como os detalhes envolvidos na sua coleta. Para isso, na Seção 3.2, o sistema operacional investigado é apresentado, assim como as justificativas de sua utilização.

A Seção 3.3 descreve o mecanismo de coleta das falhas de SO. As Seções 3.4 e 3.5 apresentam o conjunto de dados do qual foi retirada a amostra de dados de falha de SO analisada nesse trabalho. Tanto o conjunto principal quanto análises realizadas sobre ele são apresentados com o intuito informativo de descrever de onde foram retirados os dados usados neste estudo. Destaca-se que todas as análises apresentadas nessas três seções foram realizadas em trabalhos anteriores do grupo de pesquisa no qual este trabalho foi realizado, portanto, não representam resultados obtidos nesta pesquisa.

Na Seção 3.6 tem-se a descrição da filtragem usada para obter uma amostra do conjunto completo. Esta amostra é o conjunto de dados usado no estudo apresentado nesta dissertação. Apesar de não ser o objetivo principal desse trabalho, o qual visa detectar padrões de falhas de SO, realizou-se também uma caracterização e análises preliminares sobre esta amostra.

3.2 Sistema Operacional Investigado

Este trabalho realizou um estudo exploratório sobre a existência de padrões de falhas em um SO de propósito geral. Atualmente, sistemas operacionais de propósito geral são utilizados por diversos tipos de sistemas computacionais, inclusive por sistemas com requisitos de alta confiabilidade. De acordo com [Li *et al.* 2008], a empresa ABB, responsável pelo fornecimento de soluções integradas de energia e automação para usinas convencionais e renováveis de geração de energia [ABB 2016], assim como a marinha dos Estados Unidos da América, possuem programas de computador, de natureza crítica, que executam sob um sistema operacional convencional, o Microsoft Windows.

A seguir é apresentado o sistema operacional de propósito geral utilizado neste trabalho e a justificativa de sua utilização.

Diferentes pesquisas (ex. [NetMarketShare 2015] e [StatCounter Global Stats 2016]) sobre sistemas operacionais de mercado mostram que o Win7 é atualmente o sistema operacional computacional mais utilizado. Esse sistema operacional é

comumente utilizado em computadores *desktops*, *notebooks*, incluindo *netbooks* e *Tablets PCs*, e estações de trabalho [Shelly *et al.* 2010].

De acordo com [NetMarketShare 2015], o Win7 possui 55% de participação no mercado, seguido pelo Windows XP (10%), Windows 8.1 (10%) e Windows 10 (9%).

Segundo [StatCounter Global Stats 2016], o novo sistema operacional da Microsoft, Windows 10, aumentou sua participação no mercado de 10% em dezembro de 2015 para 12% em janeiro de 2016, se tornando o segundo sistema operacional de proposito geral com maior participação, seguido pelo Windows 8.1 com 10%. Apesar do aumento observado para o Windows 10, o Win7 continua prevalecendo com 42% de participação.

Conforme apresentado em [Endler 2015], a descontinuidade do suporte ao Windows XP, em 2014, favoreceu o Win7, aumentando sua participação no mercado para quase 50%, seguido pelos outros sistemas da família Windows.

Além de ser o sistema operacional mais usado atualmente, o Win7 fornece registros detalhados de falhas por meio do *Reliability Analysis Component* (RAC) [Microsoft 2016a]. O RAC é uma infraestrutura em nível de *Kernel* especializada na captura e armazenamento de dados de confiabilidade do Windows. Ele registra eventos tais como o resultado de instalações e atualizações de *software*, falhas de aplicação e do sistema operacional, e problemas provenientes do *hardware* [Microsoft 2016b]. Considerando os diferentes tipos de eventos que são armazenadas pelo RAC, esta pesquisa focou na análise das falhas de SO. O RAC é ativado automaticamente durante a instalação do sistema operacional. Portanto, este contém todos os eventos de falha de SO registrados desde a data de instalação do Win7 em um computador.

Por ser o sistema operacional com a maior participação no mercado e por possuir um componente especializado em coleta de dados de falha, o Win7 foi escolhido como o sistema alvo desta pesquisa. A Seção 3.3 explica como os registros de falha do RAC são armazenados no Win7 e o mecanismo que foi utilizado para coletá-los.

3.3 Mecanismo de Coleta de Dados

Os registros de falha do Win7, criados pelo RAC, são armazenados em um arquivo SDF (SQL *Compact Edition Database File*). Todo computador executando este sistema operacional possui um arquivo SDF para esse propósito.

Foram utilizados dois métodos para coletar os arquivos SDFs dos computadores analisados. No primeiro método, os SDFs foram copiados manualmente. Portanto, foi necessário ter acesso ao local onde os computadores estavam. Foi criado um formulário para caracterizar cada computar pesquisado. Essencialmente, o formulário apresenta informações do ambiente de trabalho e do perfil de utilização dos computadores, ou seja, as aplicações mais utilizadas. O segundo método utilizou um repositório *online* para receber os arquivos SDFs por meio de uma página *web* [HPDCS 2016a]. O mesmo formulário de caracterização dos computadores foi preenchido para cada SDF recebido. As Figuras 1.A e 2.A, no Apêndice, apresentam imagens da página *web*. A primeira exibe o formulário de

caracterização dos computadores e a segunda mostra as instruções de envio do arquivo SDF.

Os dados coletados pelo RAC também são utilizados para gerar o índice de estabilidade do computador por meio do monitor de confiabilidade (RM - *Reliability Monitor*), um programa que faz parte da distribuição do Win7. O RM [Microsoft 2016c] é uma ferramenta utilizada para mensurar problemas de *hardware* e de software e outras alterações no computador. Dessa forma, essa ferramenta fornece o índice de estabilidade do computador, o qual é um valor de 0 a 10. Quanto maior esse índice, mais estável está o sistema [Microsoft 2016d].

A Figura 3.1 mostra a tela principal do RM. Os dados coletados pelo RAC são visualizados por meio de um gráfico. A linha representa a curva de estabilidade e os círculos representam os eventos que influenciam o cálculo do índice. Pode-se verificar que a falha do programa Windows Explorer, que ocorreu em 28/11/2015, afetou diretamente o índice de estabilidade. Ao clicar sobre um evento específico é possível obter detalhes sobre o mesmo.

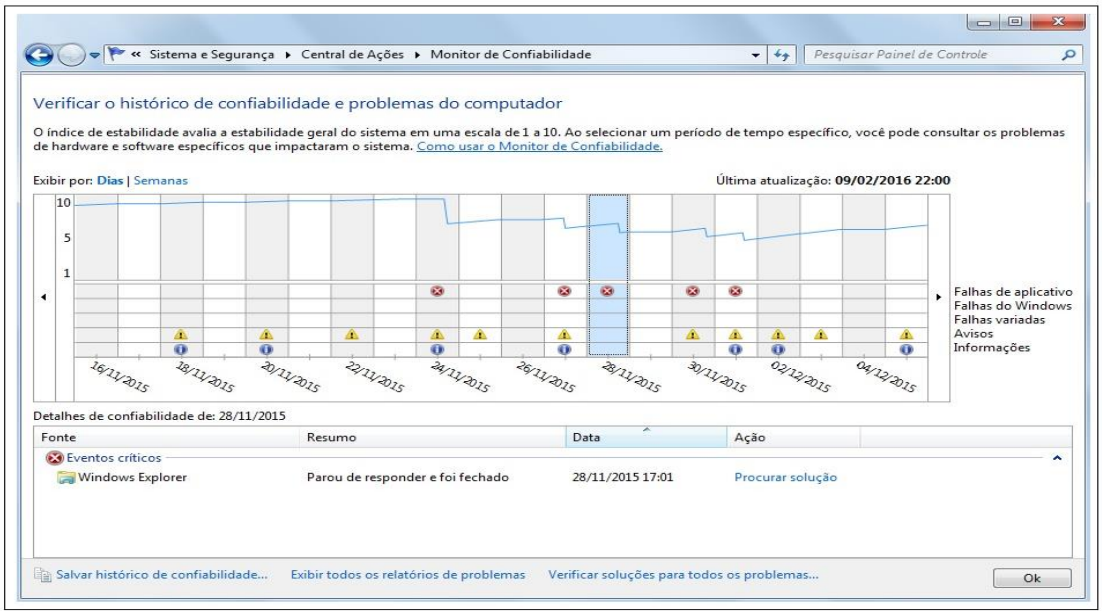


Figura 3.1 Monitor de confiabilidade (*Reliability Monitor*).

Como os arquivos SDFs não armazenam apenas falhas de SO, uma aplicação capaz de ler e filtrar seu conteúdo foi desenvolvida para ser usada nos trabalhos [Matias *et al.* 2013], [Matias *et al.* 2014] e [Antunes e Matias 2014]. Essa aplicação foi programada na linguagem Java. Esta linguagem foi escolhida por ter portabilidade entre diversos sistemas operacionais, ou seja, a aplicação pode ser executada em qualquer sistema operacional que suporte Java, sem alterações. Atualmente, uma nova aplicação, mais sofisticada, está sendo desenvolvida para esse propósito. Trata-se da OSRat (*Operating System Reliability Analysis Tool*) [HPDCS 2016b], uma ferramenta voltada para o processamento de dados de falha de sistemas operacionais. Atualmente, a OSRat suporta apenas a leitura de arquivos do RAC, no Win7, contudo seu projeto prevê novas extensões para outros sistemas, tais como outras versões do Windows, e outros sistemas operacionais tais como o Linux e o OS X.

Para todo evento capturado pelo RAC um registro de dez campos é criado [Microsoft 2016e]. A seguir uma descrição destes campos:

- ***ComputerName*:** contém o nome do computador que gerou o evento.
- ***EventIdentifier***: especifica o identificador do evento. O valor deste campo é específico da fonte que gerou o evento.
- ***InsertionStrings***: armazena informações adicionais sobre o evento.
- ***Logfile***: especifica o nome do arquivo de log do evento.
- ***Message***: apresenta uma mensagem descritiva do evento, da mesma forma que é apresentada no log do sistema.
- ***ProductName***: identifica o produto de software que estava associcado com o evento.
- ***RecordNumber***: contém o número do registro do evento armazenado no arquivo de log do sistema.
- ***SourceName***: especifica o nome da fonte (aplicação, serviço ou subsistema do *Kernel*) que gerou o evento.
- ***TimeGenerated***: descreve a hora e a data do evento, isto é, o momento em que o evento foi percebido e registrado pelo RAC.
- ***User***: identifica o usuário do sistema no momento da ocorrência do evento.

Como já destacado na Seção 2.2.1, todos os registros de falha de SO coletados foram classificados em uma das três categorias (Aplicação de SO, Serviço de SO e *Kernel*) propostas em [Matias *et al.* 2013]. Inicialmente, foi realizada uma classificação manual das falhas de SO de uma parte do conjunto de dados original, chamada de amostra piloto. Com base nessa amostra piloto foi possível automatizar o processo de classificação das falhas de SO do restante do conjunto de dados utilizando a aplicação descrita anteriormente. A classificação se baseou, principalmente, em três campos do registro de falhas do RAC: *EventIdentifier*, *SourceName* e *ProductName*. A Tabela 3.1 mostra um exemplo de como essa classificação foi realizada.

Tabela 3.1. Exemplo de classificação das falhas de SO

EventIdentifier	***SourceName***	***ProductName***	**Categoria**
20	Serviço *WindowsUpdate*	Atualização do MS Office	Serviço de SO
1000	*Application Error*	explorer.exe	Aplicação de SO
1000	*Application Error*	firefox.exe	*Não Aplicado*
1002	*Application Hang*	mmc.exe	Aplicação de SO
1137	*Windows StartupRepair*	Windows	*Kernel*

O campo *EventIdentifier* contém um valor numérico que identifica a fonte que gerou o evento. Este campo é utilizado em conjunto com o campo *SourceName* para identificar exclusivamente um tipo de evento do Windows [Microsoft 2016c]. No entanto, notou-se que esses campos não são suficientes para distinguir aplicações de SO e de usuário. Assim um terceiro campo (*ProductName*) também foi analisado. Na Tabela 3.1, dois eventos de falha, realçados pelas linhas sombreadas,

possuem o *EventIdentifier* e *SourceName* iguais a 1000 e "*Application Error*", respectivamente; nestes casos ambos são analisados com relação ao campo *ProductName*. Por meio deste campo é possível identificar que o segundo evento está relacionado com o programa *firefox.exe*, que é uma aplicação de usuário e não uma aplicação de SO. Em contraposição, o primeiro evento está relacionado com o *explorer.exe*, que é uma aplicação de SO.

A seguir tem-se a descrição de como as falhas de sistemas operacionais foram agrupadas.

3.4 Agrupamentos

O Conjunto de dados original, utilizado em trabalhos anteriores do mesmo grupo de pesquisa no qual este trabalho foi realizado, possui 30.815 falhas de SO obtidas de 735 computadores executando o sistema operacional Win7. Deste conjunto de dados, retirou-se uma amostra que foi usada neste trabalho (ver Seção 3.6). No entanto, com o propósito de apresentar uma visão geral do conjunto completo de falhas de SO, de onde foi retirada a amostra usada neste estudo, esta seção apresenta a caracterização desses dados realizada em trabalhos anteriores.

Em [Matias *et al.* 2014] os registros de falha de SO, coletados de 735 computadores, foram organizados em seis grupos (G1 até G6). Porém, para diferenciar da nomenclatura utilizada na amostra de trabalho utilizada nesta dissertação (Seção 3.6), esses seis grupos serão daqui em diante tratados como **Ga**, **Gb**, **Gc** até **Gf**, sendo que o grupo **Ga** na descrição a seguir corresponde ao grupo G1 de [Matias *et al.* 2014], **Gb** ao G2 e assim por diante. Os arquivos SDFs dos cinco primeiros grupos foram coletados manualmente. No sexto grupo, os SDFs foram coletados por meio do formulário *online* (ver Seção 3.3). A Tabela 3.2 exibe o perfil de uso dos computadores de cada grupo.

Tabela 3.2. Perfil de uso dos grupos investigados

Grupos	Ambiente	Perfil de Aplicação
Ga	Acadêmico	Aplicativos de escritório, edição gráfica.
Gb	Acadêmico	Aplicativos de escritório, IDE de programação, multimídia, ciência e engenharia.
Gc	Acadêmico	Aplicativos de escritório, edição gráfica e IDE de programação.
Gd	Corporativo	Aplicativos de escritório, edição gráfica e IDE de programação.
Ge	Corporativo	Aplicativos de escritório, edição gráfica e IDE de programação.
Gf	Acadêmico, Corporativo, Pessoal.	Aplicativos de escritório, edição gráfica, IDE de programação, multimídia, ciência e engenharia, jogos, aplicativos ERP, antivírus, servidores: banco de dados, *web*, aplicações, e-mail, diretórios e arquivo/impressão.

Fonte: Traduzido de [Matias *et al.* 2014]

Os grupos **Ga** e **Gb** contêm falhas em computadores de duas universidades diferentes. Em ambos os grupos, os computadores são utilizados por estudantes de graduação.

Os dados de falha dos grupos **Gc** e **Gd** são provenientes de computadores de uma mesma universidade. **Gc** contém falhas de computadores de laboratórios de ensino de cursos de graduação, com perfil de uso similar ao dos grupos **Ga** e **Gb**. Diferentemente, o grupo **Gd** reúne registros de falha coletados de computadores do departamento administrativo da universidade.

O grupo **Ge** possui registros de falha coletados de computadores de um ambiente corporativo. O **Gf** é o único grupo heterogêneo, ou seja, contém registros de falha de SO de diferentes ambientes de trabalho, variando de corporativo, acadêmico e computadores pessoais.

A Tabela 3.3, traduzida de [Matias *et al.* 2014], mostra a porcentagem de falhas de SO presentes em cada categoria considerada (Aplicação de SO, Serviço de SO e *Kernel*) com relação aos ambientes analisados (**Ga**, **Gb**, **Gc**, **Gd**, **Ge** e **Gf**). Nota-se que a categoria Serviço de SO (SO_{SVC}) apresentou o maior percentual das falhas de SO, enquanto o percentual de falhas nas categorias *Kernel* (SO_{KNL}) e Aplicação de SO (SO_{APP}) foram menores e similares.

Tabela 3.3. Categorização das falhas de SO por grupo

	Ga	**Gb**	**Gc**	**Gd**	**Ge**	**Gf**
Período de amostragem (Dias)	269	72	332	391	378	891
Total de computadores	5	53	268	275	41	83
Total de falhas de SO	284	406	6.844	19.725	548	3.008
$\%SO_{APP}$	0,35	3,94	0,89	20,55	9,87	5,65
$\%SO_{SVC}$	97,89	92,61	84,07	76,92	82,27	85,14
$\%SO_{KNL}$	1,76	3,45	15,04	2,53	7,86	9,21
Falhas por dia (Média)	1,06	5,64	20,61	50,45	1,45	3,38

Fonte: Traduzido de [Matias *et al.* 2014]

3.5 Propriedades Estatísticas

O conjunto de dados original foi analisado estatisticamente em trabalhos anteriores, cujos principais resultados são sumarizados nesta seção.

Em [Matias *et al.* 2013] e [Matias *et al.* 2014], os tempos das ocorrências das falhas, isto é, os registros do RAC nos momentos em que as falhas ocorreram, foram analisados estatisticamente por meio de testes de aderência (*goodness-of-fit*).

Os testes de aderência medem o quanto uma determinada distribuição teórica se assemelha com os dados [Rayner 2009]. Portanto, os testes de aderência foram utilizados, pois permitem selecionar a distribuição que melhor se adequa aos dados, a fim de usá-la para estimar as métricas de confiabilidade de interesse.

Em [Matias *et al.* 2014], foram adotados quatro testes de aderência amplamente usados: Anderson Darling [Cullen e Frey 1999], Kolmogorov-Smirnov [Cullen e Frey 1999], Qui-Quadrado [Conover 1999] e Razão de verossimilhança [Mood *et. al* 1974]. O método de estimação de parâmetros usado foi o método de máxima verossimilhança. Nem todos os quatro testes foram aplicados a todos os dados, já que isso depende dos requisitos de cada teste. De acordo com os autores, não é incomum ocorrer resultados conflitantes entre os testes para um mesmo conjunto de dados devido às especificidades de cada algoritmo. Portanto, os resultados foram comparados e aqueles que geraram maior concordância foram selecionados.

Os autores de [Matias *et al.* 2014] optaram por aplicar os testes de aderência com distribuições amplamente usadas em estudos de confiabilidade. Quinze distribuições de probabilidade foram consideradas: Exponencial, Exponencial com 2 parâmetros, Gama, Gama generalizada, Gumbel, Maior Valor Extremo, Logística, Log-logística, Log-logística com 3 parâmetros, Lognormal, Lognormal com 3 parâmetros, Normal, Menor Valor Extremo, Weibull e Weibull com 3 parâmetros.

A Tabela 3.4 apresenta os resultados obtidos em [Matias *et al.* 2014] dos testes de aderência aplicados no conjunto de dados original apresentado anteriormente (ver Seção 3.4). Nota-se que, em geral, as distribuições Gama e Weibull apresentaram a melhor qualidade de ajuste nas categorias, sendo a Gama prevalente para serviços de SO e a Weibull para aplicações de SO e *Kernel*. De acordo com os autores, as quatro entradas vazias apresentaram conjunto de dados muito pequenos de falhas de SO, inviabilizando a identificação de uma distribuição.

Tabela 3.4. Distribuições com melhor ajuste por categoria de falha e por grupo

Grupos	Aplicação de SO	Serviço de SO	*Kernel*
Ga	-	Gama (k=0,12; μ=6,11)	-
Gb	-	Weibull (β=0,82; η=279)	-
Gc	Weibull (β=0,32; η=264)	Gama (k=0,16; μ=7,50)	Gama (k=0,45; μ=6,81)
Gd	Log-logística (μ=-6,07; σ=2,1)	Gama (k=0,14; μ=6,64)	Weibull (β=0,32; η=254)
Ge	Weibull (γ=0; β=0,33; η=382)	Gama (k=0,35; μ=6,45)	Weibull (β=0,51; η=254)
Gf	Weibull (β=0,44; η=482)	Gama (k=0,30; μ=6,71)	Log-logística (μ=4,26; σ=1,1)

Fonte: Traduzido de [Matias *et al.* 2014]

Com o propósito de obter uma visão geral da confiabilidade, os autores também aplicaram os testes de aderência por grupo. Como resultado, a Gama foi a distribuição que melhor se ajustou aos dados. Segundo os autores, isso ocorreu, pois a distribuição Gama obteve a melhor qualidade de ajuste para a categoria Serviço de SO e esta categoria possui a maior quantidade de falhas em todos os grupos do conjunto de dados original (ver Tabela 3.3).

Com base nas distribuições com os melhores ajustes aos tempos de ocorrência das falhas de SO do conjunto de dados, os autores calcularam um conjunto de métricas de confiabilidade, as quais estão listadas na Tabela 3.5.

Tabela 3.5. Métricas de confiabilidade

Métrica	Descrição
Confiabilidade $R(t)$	Probabilidade que o sistema irá funcionar corretamente até o tempo t
Probabilidade de Falha $F(t) = 1 - R(t)$	Probabilidade que o sistema irá falhar após o tempo t
Taxa de Falha $\lambda(t)$	Taxa de falha dado um tempo de sobrevivência t
MTBF	Tempo médio entre falhas
Tempo de Garantia $t(R)$	Momento no qual um valor de confiabilidade específico será alcançado
Vida Bx%	A hora em que x% dos sistemas da amostra terão falhado

Fonte: Traduzido de [Matias *et al.* 2014]

As três primeiras métricas da Tabela 3.5 são bastante utilizadas em engenharia de confiabilidade [Trivedi 2001]. Com base na R(t), é possível estimar a quarta métrica, MTBF, que é o tempo entre duas falhas subsequentes de SO. Esta é uma métrica fundamental para confiabilidade em sistemas reparáveis. A quinta métrica indica o tempo de garantia t para uma dada confiabilidade. Por exemplo, até quando (t) um sistema garante sua operação com um nível de confiabilidade de 0,90 (R=0,90). Os autores utilizaram o valor alvo R = 0,95 para a função tempo de garantia. A última métrica mostra o momento em que x% dos sistemas terão falhado. Por exemplo, uma vida B10% de 70h significa que 10% dos sistemas terão falhado após 70 horas de operação. Foi usado o valor alvo 10% para a vida Bx%.

As Tabelas 3.6, 3.7 e 3.8 exibem a estimação dessas métricas de confiabilidade para o conjunto de dados original realizada em [Matias *et al.* 2014]. Segundo os autores, em alguns casos, os valores alvo para $t(R)$ e para a função Bx% não foram encontrados devido aos dados coletados. Por exemplo, na Tabela 3.6, não foi possível alcançar o valor alvo de $t(R$=0,95) no grupo **Gd**, devido a pouca confiabilidade observada nos sistemas desse grupo.

Tabela 3.6. Métricas de confiabilidade para Aplicação de SO

	Ga	Gb	Gc	Gd	Ge	Gf
$R(t$=8h)	-	-	0,722	0,164	0,758	0,847
$F(t$=8h)	-	-	0,278	0,836	0,242	0,152
$\lambda(t)$	-	-	0,013/h	0,026/h	0,011/h	0,009/h
MTBF (h)	-	-	1825,88	35,43	2314,03	1265,05
$t(R)$ horas	-	-	R=0,95 0,025h	R=0,55 1,9E-21h	R=0,95 0,053h	R=0,95 0,556h
Bx% horas	-	-	x=10 0,240h	x=45 1,9E-21h	x=10 0,441h	x=10 2,864h

Fonte: Traduzido de [Matias *et al.* 2014]

Tabela 3.7. Métricas de confiabilidade para Serviço de SO

	Ga	Gb	Gc	Gd	Ge	Gf
$R(t$=8h)	0,344	0,947	0,218	0,414	0,739	0,415
$F(t$=8h)	0,656	0,053	0,782	0,586	0,261	0,585
$\lambda(t)$	0,027/h	0,006/h	0,017/h	0,022/h	0,014/h	0,014/h
MTBF (h)	53,730	311,123	114,518	102,384	216,784	142,082
$t(R)$ horas	R=0,95 1,9E-11h	R=0,95 7,380h	R=0,40 5,0E-12h	R=0,92 5,9E-12h	R=0,95 0,011h	R=0,57 0,000h
Bx% horas	x=10 1,8E-07h	x=10 17,796h	x=60 5,0E-12h	x=10 1,7E-08h	x=10 0,291h	x=43 0,000h

Fonte: Traduzido de [Matias *et al.* 2014]

Tabela 3.8. Métricas de confiabilidade para *Kernel*

	Ga	Gb	Gc	Gd	Ge	Gf
$R(t$=8h)	-	-	0,868	0,708	0,841	0,880
$F(t$=8h)	-	-	0,132	0,292	0,159	0,120
$\lambda(t)$	-	-	0,009/h	0,013/h	0,011/h	0,014/h
MTBF (h)	-	-	411,709	1668,893	493,424	1,0E+99
$t(R)$ horas	-	-	R=0,95 0,928h	R=0,95 0,006h	R=0,95 0,732h	R=0,95 2,834h
Bx% horas	-	-	x=10 4,304h	x=10 0,133h	x=10 3,021h	x=10 6,412h

Fonte: Traduzido de [Matias *et al.* 2014]

Em geral, os resultados obtidos corroboram a análise de caracterização realizada (ver Tabela 3.3). Para 8 horas de funcionamento, apenas dois grupos apresentaram confiabilidade, R(t), superior a 50% na categoria Serviço de SO. Além disso, os autores de [Matias *et al.* 2014] compararam o $R(t$=8) com outras métricas, como o MTBF, dado que nem sempre a confiabilidade mais alta apresenta o maior MTBF. Por exemplo, na Tabela 3.6, o **Gf** apresenta a maior confiabilidade durante 8 horas de operação, mas o **Gc** apresenta um MTBF significativamente maior. De maneira semelhante, na Tabela 3.8, o **Gc** apresenta maior confiabilidade durante 8 horas do que no **Gd**, mas o MTBF deste é mais de quatro vezes maior do que o MTBF do **Gc**.

As métricas de confiabilidade também foram estimadas por grupo. Essa estimação foi realizada por meio da distribuição Gama, que foi distribuição que melhor se ajustou aos dados quando os testes de aderência foram aplicados por grupos. A Tabela 3.9 apresenta esta estimação. De acordo com os autores, uma vez que as métricas de confiabilidade estimadas por grupo consideram todas as categorias de falha de SO, os resultados oferecem uma visão geral da confiabilidade dos sistemas dos grupos. Assim, eles concluíram que os sistemas de **Gb** e **Ge** são os

mais confiáveis. Esta mesma conclusão pode ser tirada a partir dos valores de MTBF.

Tabela 3.9. Métricas de confiabilidade por grupo

	Ga	Gb	Gc	Gd	Ge	Gf
$R(t$=8h)	0,345	0,921	0,294	0,357	0,743	0,458
$F(t$=8h)	0,655	0,079	0,706	0,643	0,257	0,542
$\lambda(t)$	0,028/h	0,007/h	0,017/h	0,023/h	0,014/h	0,015/h
MTBF (h)	52,633	282,338	101,205	82,454	187,455	127,640
$t(R)$	R=0,95	R=0,95	R=0,49	R=0,85	R=0,95	R=0,63
horas	3,7E-11h	3,978h	0,000h	7,3E-15h	0,025h	0,000h
Bx%	x=10	x=10	x=51	x=15	x=10	x=37
horas	2,7E-07h	11,452h	0,000h	7,3E-15h	0,398h	0,000h

Fonte: Traduzido de [Matias *et al.* 2014]

Além das métricas de confiabilidade descritas na Tabela 3.9, os autores calcularam as curvas gerais de confiabilidade para os seis grupos. A Figura 3.2 apresenta essas curvas. Eles observaram que a maioria das falhas de **Gb** diminuiu depois do período de 8 horas usado no cálculo de $R(t)$ (ver Tabela 3.9), o que explica sua alta confiabilidade em relação aos demais grupos. Comportamento semelhante foi observado no **Ge**, que também apresenta estimativa consistente em todas as categorias de falha analisadas. Assim, de acordo com os autores, **Gb** e **Ge** apresentam, respectivamente, a melhor experiência para o usuário com relação à confiabilidade de SO de acordo com os ambientes estudados.

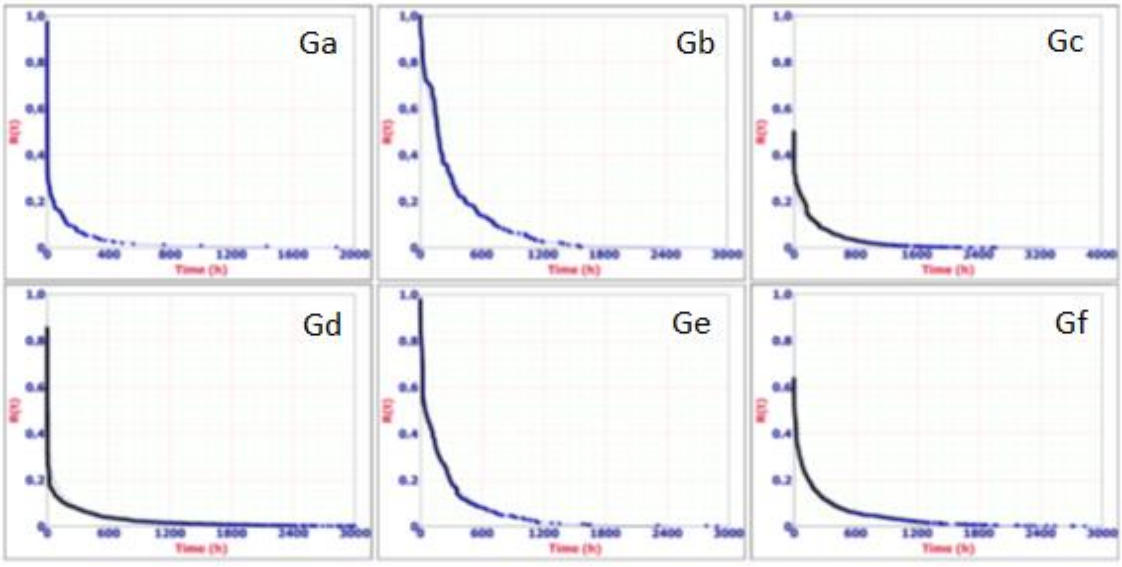


Figura 3.2. Curva de confiabilidade por grupo.
Fonte: Traduzido de [Matias *et al.* 2014]

Por fim, os autores de [Matias *et al.* 2014] tiveram as seguintes conclusões.

- As falhas de serviços de SO prevaleceram em relação as falhas de aplicações de SO e *Kernel*, que tiveram quantidades similares.

- As falhas de *Kernel* do sistema operacional foram mais prevalentes em ambientes corporativos do que em ambientes acadêmicos. Segundo os autores, isso pode estar relacionado com o tempo de execução dos sistemas. Considerando que o tempo médio de operação do SO em laboratórios de ensino deva ser menor do que em ambientes corporativos, maiores são as chances de falhas de *Kernel* ocorrerem neste último, visto que a maior duração de operação ininterrupta do sistema aumenta as chances de propagações de erro no *Kernel*.
- Os resultados obtidos com base em dados reais proporcionaram as distribuições de probabilidade que melhor se ajustaram aos dados. Para os autores, este conhecimento pode ser utilizado em diferentes estudos na área de modelagem e simulação de confiabilidade de sistemas de computação.

A próxima seção apresenta a amostra de trabalho utilizada nesta pesquisa, extraída do conjunto original de dados de falha descrito na seção anterior.

3.6 Amostra de Trabalho

Nas seções anteriores foi apresentada a caracterização geral do conjunto completo de dados de falha de SO usado em trabalhos anteriores do grupo de pesquisa no qual este trabalho foi realizado. Em virtude das características e objetivos específicos desta pesquisa, o presente trabalho utilizou uma amostra do conjunto de dados descrito anteriormente.

Diversos registros de falha do conjunto original foram removidos para compor a amostra de trabalho deste estudo, isto por serem considerados *outliers* para o objetivo deste trabalho, o qual propõe investigar a existência de padrões de falhas de SO. Neste contexto, é importante que o mesmo tipo de falha de SO se repita em diferentes ambientes de trabalho, de forma sistemática, o que não ocorre com todos os registros.

Com esse propósito, o conjunto original foi examinado e avaliou-se não apenas o número de ocorrências do mesmo evento de falha, mas também se o mesmo evento de falha foi observado em vários computadores de diferentes ambientes de trabalho (grupos). Para este trabalho, quanto mais um evento de falha é observado em diferentes computadores e grupos, maior será sua consistência e, consequentemente, a sua importância para o estudo. Assim, o conjunto original foi filtrado para remover todos os casos em que a mesma falha não foi observada diversas vezes em diferentes computadores de diferentes grupos, incluindo falhas em vários computadores, mas com prevalência em um único.

Os grupos **Ga** e **Gb** do conjunto original foram removidos, pois apresentaram amostras muito pequenas de falhas de SO. Como resultado, a amostra de trabalho ficou composta de 7.007 falhas de SO obtidas de 566 computadores. Os registros das falhas foram organizados em quatro grupos (G1, G2, G3 e G4) que, na descrição do trabalho anterior ([Matias *et al.* 2014]), na Seção 3.4, foram denominados grupos **Gc**, **Gd**, **Ge** e **Gf**.

Os dados de falha analisados foram capturados e armazenados pelo RAC durante um período de cinco anos. A Tabela 3.10 apresenta os períodos de

captura e armazenamento das falhas de SO de cada grupo. O único grupo que mais falhas de SO foram incluídas a ele, foi o grupo G4, anteriormente **Gf**. Os períodos exibidos na Tabela 3.10 da amostra de trabalho diferiram dos períodos do conjunto de dados original. Isso ocorreu, pois este período é calculado em relação aos tempos de ocorrência dos registros de falha, como muitas falhas foram removidas e outras adicionadas, o período também foi modificado.

Tabela 3.10. Período de amostragem por grupo

Grupos	Período de amostragem
G1	13/12/2011 to 12/11/2012
G2	24/10/2011 to 19/11/2012
G3	24/10/2011 to 05/11/2012
G4	26/09/2010 to 08/23/2014

Todas as falhas de sistema operacional foram classificadas como aplicações de SO (SO_{APP}), serviços de SO (SO_{SVC}) e *Kernel* (SO_{KNL}) na amostra de trabalho. Nessa amostra, tem-se novamente que a categoria Serviço de SO foi responsável pelo maior percentual de falhas na maioria dos grupos. As falhas de *Kernel* apresentaram os menores percentuais, tendo valores similares aos da categoria Aplicação de SO. A Figura 3.11 exibe uma representação da classificação realizada. A Tabela 3.11 mostra um resumo dos dados de falha de SO utilizados neste estudo.

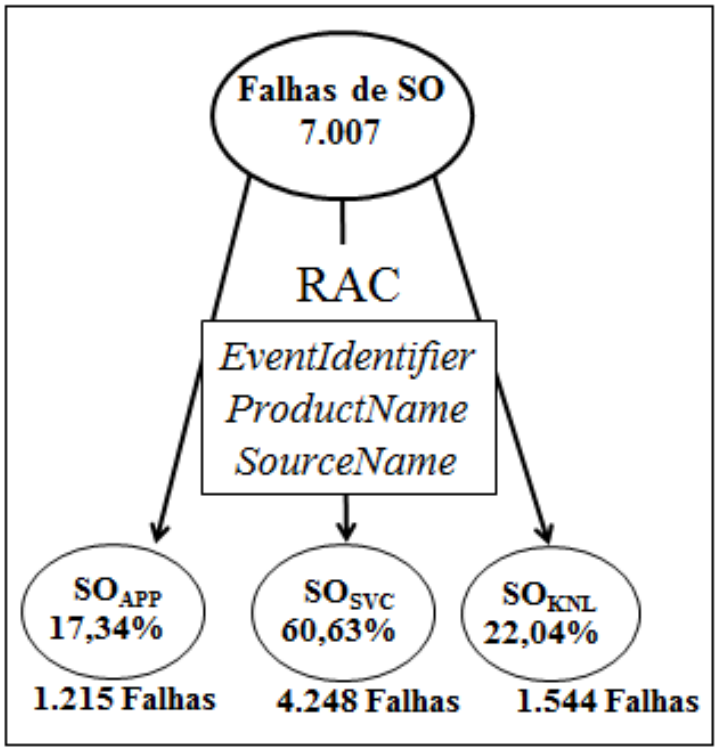


Figura 3.3. Categorização das falhas de SO.

Tabela 3.11. Resumo da amostra de falhas de SO

	G1	G2	G3	G4	Total
Período de amostragem (dias)	331	564	360	1.288	2.543
Total de computadores	229	211	29	97	566
Total de falhas de SO	1.090	3.445	306	2.166	7.007
% Aplicação de SO	5,50	11,70	18,63	32,09	22,04
% Serviço de SO	10,37	77,30	69,93	58,08	60,63
% Kernel	84,13	11,00	11,44	9,83	17,34
Falhas de SO normalizadas pelo total de computadores	4,76	16,33	10,55	22,33	12,38

Apenas no grupo G1, as falhas de *Kernel* predominaram em relação às falhas das outras categorias. Isso pode indicar a influência de fatores ambientais (ex. configurações dos computadores e cargas de trabalho), presentes neste grupo, os quais contribuíram para provocar essas falhas. Verificou-se que os computadores com maior quantidade de falhas nas três categorias se concentraram nos grupos G2 e G4.

Os dados de falha também foram caracterizados em relação ao número de ocorrências de falhas de SO por período do dia. Os períodos considerados foram: Madrugada (00:00 – 06:59), Manhã (7:00 – 12:59), Tarde (13:00 – 18:59) e Noite (19:00 – 23:59). A Tabela 3.12 apresenta os resultados obtidos. As áreas sombreadas indicam os períodos do dia com maior número de falhas de SO por grupo.

No geral, baseado nos campos “Todos os Grupos” e “Todas as Categorias” da Tabela 3.12, observou-se um alto número de falhas de SO no período da tarde. As falhas de *Kernel* ocorreram mais frequentemente no período da noite, enquanto as falhas de aplicações de SO foram mais frequentes no período da manhã e as falhas de serviços de SO no período da tarde.

Apenas no grupo G3 as falhas da categoria Serviço de SO foram observadas com mais frequência no período da madrugada. Neste caso específico, observou-se que muitas dessas falhas ocorreram durante as atualizações de *software* realizadas pelo serviço Windows *Update* (SWU) às 03:00. Este serviço é o mecanismo padrão para atualizar *software* no sistema operacional Win7. Essas falhas do SWU no G3, observadas durante o período da madrugada, sugerem que o serviço Windows *Update* estava programado para executar suas operações automaticamente, pois, de acordo com [Boyce 2009], 03:00 é o horário padrão para instalar novas atualizações de *software* quando este serviço está definido para ser executado em modo automático.

Tabela 3.12. Percentual de falhas por período do dia

Grupos	Período do Dia	*Kernel* (%)	Aplicação de SO (%)	Serviço de SO (%)	Todas as Categorias (%)
G1	Madrugada	0	0	0	0
	Manhã	12,10	6,67	38,94	14,59
	Tarde	22,03	30,00	16,81	21,93
	Noite	65,87	63,33	44,25	63,49
G2	Madrugada	0,53	0	0,26	0,26
	Manhã	46,97	56,33	29,22	34,34
	Tarde	45,91	39,95	58,24	54,75
	Noite	6,60	3,72	12,28	10,65
G3	Madrugada	0	5,26	44,89	32,35
	Manhã	60,00	47,37	33,18	38,89
	Tarde	28,57	45,61	20,09	25,82
	Noite	11,43	1,75	1,87	2,94
G4	Madrugada	10,33	4,32	19,16	13,53
	Manhã	20,66	38,99	26,87	30,15
	Tarde	41,78	38,56	31,08	34,53
	Noite	27,23	18,13	22,89	21,79
Todos os Grupos	Madrugada	1,55	2,72	8,10	5,72
	Manhã	22,93	43,54	28,98	30,17
	Tarde	30,76	38,93	47,18	42,13
	Noite	44,75	14,81	15,75	22,98

3.6.1 Tipos e Subtipos de Falha de Sistema Operacional

Todas as falhas de sistema operacional classificadas anteriormente como aplicações de SO (SO_{APP}), serviços de SO (SO_{SVC}), e *Kernel* (SO_{KNL}) no conjunto de dados analisado (amostra de trabalho), foram subdivididas em 113 tipos de falha, de acordo com o campo *ProductName* dos registros de falha do RAC. Dos 113 tipos, 23 são de Aplicação de SO, 55 de Serviço de SO e 35 são relacionadas ao *Kernel*.

A categoria Serviço de SO contém a maior quantidade de eventos de falha (ver Tabela 3.11) e tipos de falha de SO. Com o propósito de compreender o motivo da grande incidência de falhas dessa categoria na amostra, os registros de falha dessa categoria foram examinados detalhadamente. Por meio do campo *SourceName* desses registros, observou-se que 23 dos 55 tipos de falha dessa categoria estavam relacionados com o serviço Windows *Update* (SWU).

O número de eventos de falha dos 23 tipos relacionados com o SWU corresponde a 87,97% (3.737 falhas) do número total de falhas (4.248 falhas) da categoria SO_{SVC}. Portanto, essas falhas representam 53,33% de todas as falhas de SO da amostra de trabalho. Em consequência da prevalência de falhas de SO relacionadas com o serviço Windows *Update* na amostra de trabalho, elas foram

agrupadas em 23 subtipos de falha dentro do tipo SWU, e foram analisadas separadamente. Cada subtipo do SWU representa uma atualização de *software* realizada pelo serviço Windows *Update* visando um componente específico de *software*. Por exemplo, Atualização do Internet Explorer (AIE) representa as atualizações de *software* aplicadas ao programa Internet Explorer (IE) e aos seus arquivos relacionados (ex. bibliotecas). A Tabela 3.13 lista os cinco tipos e subtipos (quando aplicável) das falhas de SO que mais ocorreram na amostra.

Tabela 3.13. Tipos e subtipos de falha de SO com maior ocorrência na amostra de trabalho

Categoria	Tipo de falha de SO [Subtipo]	% de eventos
SO_{SVC}	SWU [Atualização do Internet Explorer]	22,63%
SO_{SVC}	SWU [Atualização do Windows]	16,61%
SO_{KNL}	VIDEO_TDR_*ERROR*	12,82%
SO_{APP}	explorer.exe	8,36%
SO_{SVC}	SWU [Atualização do .Net Framework]	4,41%

Na Tabela 3.13, todas as falhas de SO_{SVC} são relacionadas com o serviço Windows *Update*. Portanto, são subtipos do tipo SWU. Nota-se que esses subtipos não representam falhas em seus respectivos *software*s (ex. Internet Explorer). Efetivamente, estão relacionados a falhas na execução do SWU quando este realiza atualizações para esses componentes de *software*. Por exemplo, o subtipo "Atualização do Internet Explorer" agrupa falhas do SWU que ocorreram enquanto o serviço Windows *Update* realizava atualizações de *software* no programa Internet Explorer.

Ao contrário dos outros subtipos de falha do SWU, que são específicos para aplicações não relacionadas com o SO, o subtipo "Atualização do Windows" (AW7) representa falhas em atualizações de *software* aplicadas exclusivamente aos componentes do sistema operacional, isto é, arquivos essenciais para o funcionamento do Win7.

Como mencionado na Seção 2.2.1, a confiabilidade do sistema operacional investigada nesta pesquisa considera a experiência do usuário, ou seja, a qualidade do serviço prestado por esse sistema ao seu usuário. Portanto, problemas no serviço Windows *Update* foram considerados como falhas de SO, pois muitas vezes prejudicam a experiência do usuário.

O serviço Windows *Update* é ativado para realizar suas funções em modo automático após a instalação do Win7. Neste caso, a experiência dos usuários pode ser afetada sem ao menos eles concordarem em receber este serviço.

Conforme apresentado na Tabela 3.13, o VIDEO_TDR_ERROR foi o tipo mais comumente observado de falha de *Kernel* na amostra. Como mencionado anteriormente (Seção 2.2.1), essa falha ocorre quando o controlador do dispositivo de vídeo detecta que já não pode mais controlar a unidade de processamento gráfico, indicando uma falha durante a tentativa de recarregar o driver de vídeo. O representante da categoria SO_{APP} foi o explorer.exe. Este programa possui duas

principais funcionalidades: gerenciar janelas e arquivos. A primeira funcionalidade implementa a tela de área de trabalho na interface gráfica do Win7, enquanto a segunda é usada para acessar e navegar em unidades de armazenamento secundárias nos níveis de arquivos e diretórios.

Analisando apenas as falhas do SWU, constatou-se que quatro subtipos foram responsáveis por 88,98% de todas as falhas relacionadas com o serviço Windows *Update*. Essas falhas estão distribuídas em 42,44% de Atualização do Internet Explorer (AIE), 31.15% de Atualização do Windows (AW7), 8.27% de Atualização do .Net Framework (ANF) e 7.12% de Atualização do MS Office (AMO).

Como esses quatros subtipos (AIE, AW7, ANF e AMO) de falha do SWU são altamente representados no conjunto de dados, seus comportamentos e causas foram analisados detalhadamente nos próximos capítulos. Foi possível analisar a causa das falhas dos subtipos por meio dos *Error Codes* presentes no campo *Message* dos registros de falha do RAC. *Error Codes* são valores hexadecimais associados com a mensagem de erro gerada durante uma falha e são usados para identificar a causa de falhas no sistema operacional MS Windows [Microsoft 2016f]. A interpretação dos *Error Codes* foi realizada por meio da documentação oficial do fabricante do SO investigado (ex. [Microsoft 2016g], [Microsoft 2016h] e [Microsoft 2016i]).

A Figura 3.4 apresenta uma visão geral da estratificação da amostra de falhas de SO. Primeiramente, as falhas foram dividias em categorias, depois em tipos, e para o tipo serviço Windows *Update* (SWU), criaram-se subtipos e, por fim, as causas das falhas foram obtidas por meio dos *Error Code*.

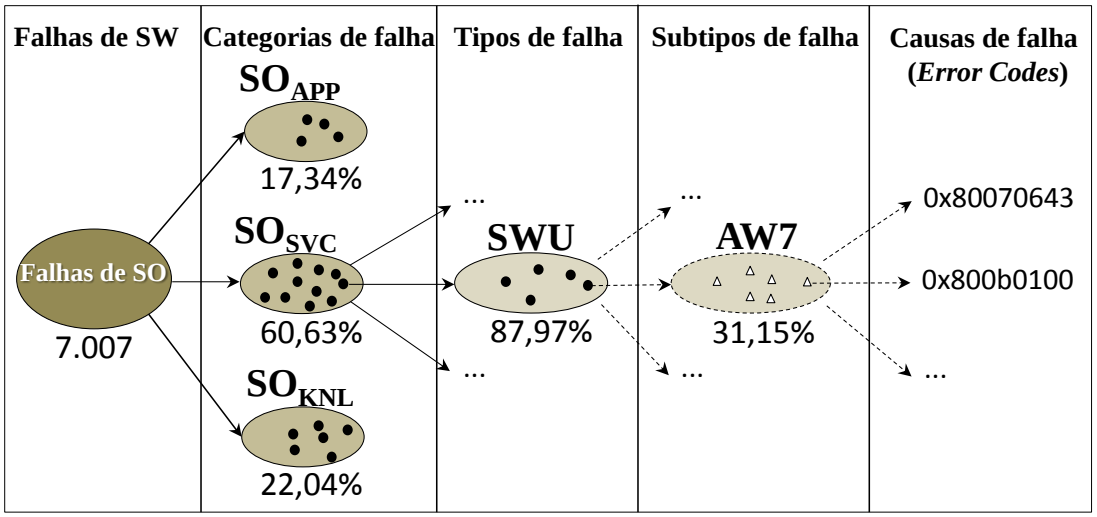


Figura 3.4. Estratificação da amostra de falhas de sistema operacional.

4. Protocolo para Detecção e Caracterização de Padrões de falhas de Sistemas Operacionais

4.1 Introdução

O propósito deste capítulo é descrever a abordagem criada, na forma de um protocolo, para detectar e caracterizar padrões de falhas de SO.

Em [Cyganek 2013], o autor apresentou métodos computacionais para seleção de objetos em imagens capturadas por sistemas ópticos. De acordo com o autor, a **detecção** frequentemente precede o **reconhecimento**. O autor define a **detecção** de um objeto como a verificação da existência do objeto na imagem analisada. Por outro lado, ele define o **reconhecimento** de um objeto como o descobrimento da categoria ou classe do objeto. Seguindo essa mesma definição, porém no contexto desta pesquisa, a **detecção** de padrões de falhas de SO foi utilizada para investigar a existência de padrões nos registros de falha de SO da amostra de trabalho, uma vez que não foram encontrados trabalhos correlatos que demonstraram a existência de tal fenômeno.

Primeiramente, na Seção 4.2, tem-se a descrição de uma taxonomia criada com o objetivo de auxiliar o desenvolvimento do protocolo.

As Seções 4.3 e 4.4 apresentam o fluxo de execução do protocolo de detecção e caracterização de falhas de SO. Neste trabalho, padrões de falhas de SO são definidos como combinações de eventos de falha que se repetem sistematicamente na amostra de trabalho. No protocolo proposto, primeiramente, as falhas de SO que ocorreram no maior número de grupos da amostra foram identificadas. Por intermédio dessas falhas, foram investigados os eventos de falha de SO que ocorreram antes e/ou depois delas, ou seja, que podem estar relacionadas com as suas causas ou efeitos.

Por fim, na Seção 4.5, tem-se a descrição da adaptação do protocolo para detectar e caracterizar padrões entre as causas de falha de SO. Essa abordagem foi aplicada, especificamente, para as falhas do serviço Windows *Update* (SWU), pertencentes à categoria Serviço de SO, pois foram essas as falhas mais abundantes na amostra de trabalho; e também por seus registros oferecerem informações sobre a causa das falhas.

4.2 Taxonomia Proposta

Devido à necessidade do desenvolvimento de uma abordagem própria para detecção e caracterização de padrões de falhas de SO, uma taxonomia própria também precisou ser criada. Essa taxonomia define vários conceitos utilizados no protocolo proposto. A seguir tem-se a descrição dos elementos dessa taxonomia.

- **Falha de referência** (**FR**): é uma falha de SO que serve como referência base para investigar padrões de falhas de SO.
- **Evento de Falha Anterior** (**EFA**): é todo evento de falha de SO que ocorre antes de uma **FR**. O intervalo de tempo entre uma **FR** e um **EFA** é chamado de **ΔtA**.
- **Evento de Falha Posterior** (**EFP**): é todo evento de falha de SO que ocorre depois de uma **FR**. O intervalo de tempo entre uma **FR** e um **EFP** é chamado de **ΔtP**.
- **Intervalo de Pesquisa** (**IP**): é o intervalo de tempo antes e/ou depois de uma **FR**, utilizado na busca pelos **EFA**s e/ou **EFP**s.
- **Padrão de Falha Candidato** (**PFC**): é uma combinação de eventos de falha de SO. São encontrados quando **EFA**s e/ou **EFP**s ocorrem juntamente com uma **FR** dentro de um determinado **IP**. O protocolo proposto considera três configurações de **PFC**s:

  PFC=**EFA**→**FR**: corresponde a uma combinação de eventos obtida a partir da análise de eventos de falha que ocorreram antes de uma falha de referência para um **IP** definido**.**

  PFC=**FR**→**EFP**: corresponde a uma combinação de eventos obtida a partir da análise de eventos de falha que ocorreram depois de uma falha de referência para um **IP** definido**.**

  PFC=**EFA**→**FR**→**EFP**: corresponde a uma combinação de eventos obtida a partir da análise de eventos de falha que ocorreram antes e depois de uma falha de referência para um **IP** definido.

  Na representação das três possíveis configurações de **PFC**s**,** as falhas de referência (**FR**s) são sempre sublinhadas e o símbolo "→" não implica necessariamente em causalidade, apenas indica a ordem temporal de ocorrência dos eventos de falha de SO pertencentes a uma dada combinação observada.

  A Tabela 4.1 lista exemplos das três configurações de **PFC**s analisadas nesse trabalho. Estes exemplos foram criados para fins didáticos e, portanto, não fazem parte dos resultados analisados neste estudo. Cada exemplo mostra combinações de eventos de falha de SO que ocorreram dentro de um intervalo de pesquisa de 2 horas antes e/ou depois das falhas de referência. Por meio da análise do campo *TimeGenerated*, que contém a data e a hora que uma falha foi registrada pelo RAC, identificou-se a ordem temporal dos eventos de falha.

  As linhas sombreadas na Tabela 4.1 destacam a combinação de eventos de falha de SO que compõem os **PFC**s, enquanto as linhas com o campo *ProductName* sublinhado indicam as falhas de referência. Todas as falhas de referência listadas são do tipo explorer.exe (Aplicação de SO).

Para a configuração **PFC=EFA→FR**, a combinação de eventos analisada é representada por **PFC**=dllhost.exe→explorer.exe. Nesse exemplo, todos os registros de falha do tipo dllhost.exe, que ocorreram antes das falhas de explorer.exe, dentro do **intervalo de pesquisa** de duas horas, são analisados. A mesma lógica se aplica aos dois outros exemplos de configuração (**PFC=FR→EFA e PFC=EFA→FR→EFP**). Apesar de nesse exemplo o caso mostrado na terceira configuração (**PFC=EFA→FR→EFP**) ter o **EFA** igual ao **EFP**, ocorrências na amostra em que ambos são diferentes também foram consideradas neste estudo; um exemplo é a combinação de eventos representada por **PFC**=Atualização do Windows→Atualização do Windows→Atualização do .Net Framework (ver Tabela 5.4).

Tabela 4.1. Exemplo de padrões de falhas candidatos

Configurações de PFCs	*ProductName*	*SourceName*	TimeGenerated
PFC=EFA→FR	dllhost.exe	Application Error	12/11/2012 09:29
	explorer.exe	Application Hang	12/11/2012 09:54
	svchost.exe	Application Error	12/11/2012 18:06
PFC=FR→EFP	rundll32.exe	Application Error	03/12/2012 06:59
	explorer.exe	Application Hang	03/12/2012 14:28
	dllhost.exe	Application Error	03/12/2012 14:46
PFC=EFA→FR→EFP	dllhost.exe	Application Error	08/06/2012 18:59
	explorer.exe	Application Hang	08/06/2012 19:03
	dllhost.exe	Application Error	08/06/2012 19:11

- **Padrões de falhas** (**PF**): é detectado quando o mesmo **PFC** se repete, de forma sistemática, em diversos computadores de diferentes grupos.
    - ***Rankings*:** Com o propósito de auxiliar a detecção de um **PF**, um ***ranking*** é criado com todos os **PFC**s ordenados pelo número de ocorrências em diferentes computadores e grupos. Quanto maior for a repetição de um **PFC** em diferentes computadores de diferentes grupos, mais alto no ***ranking*** ele estará, indicando que esse **PFC** possui evidências mais consistentes para ser considerado um **Padrão de Falha** (**PF**).

      Neste estudo, os ***rankings*** foram criados para cada intervalo de pesquisa e configuração de **PFC** (ex. ***ranking*** da configuração **PFC=EFA→FR** para o **intervalo de pesquisa** de 4 horas). A composição de um ***ranking*** obedece aos seguintes critérios, ordenados de acordo com a sua importância:

      1. Maior número de grupos que o **PFC** é observado.
      2. Maior número de computadores que o **PFC** é observado.
      3. Maior número de ocorrências do **PFC**.
      4. Menor desvio padrão de **ΔtA** e/ou **ΔtP** para a composição do **PFC**.

Portanto, os **PFC**s que forem observados em maior frequência em diferentes grupos e computadores possuem evidências mais consistentes para serem considerados **Padrões de Falhas**. No caso de um ou mais critérios empatarem, o critério subsequente é avaliado seguindo a ordem de precedência.

4.3 Protocolo de Detecção e Caracterização de Padrões

Como discutido na Seção 4.1, foi necessário desenvolver um protocolo de detecção e caracterização de padrões para a análise das falhas de SO da amostra de trabalho investigada. O protocolo identifica os eventos de falha de SO por meio dos seus tipos/subtipos e, assim, analisa combinações de eventos de falha de SO (**PFC**s) na amostra.

No protocolo proposto, primeiramente, as falhas de SO que ocorreram no maior número de grupos da amostra são identificadas. Essas falhas são utilizadas como referência para a investigação de possíveis padrões de falhas de SO, sendo chamadas de falhas de referência.

Por intermédio das falhas de referência, foram investigados os eventos de falha de SO que ocorreram antes e/ou depois delas, ou seja, que podem estar relacionadas com as suas causas ou efeitos.

Para caracterizar as ocorrências das falhas antes e/ou depois da falha de referência foram criadas três possíveis configurações de padrões de falhas. As falhas de SO podem ocorrer i) somente antes, ii) somente depois e iii) antes e depois das falhas de referência.

Os **PFC**s são encontrados quando diferentes eventos de falha de SO são observados antes e/ou depois de uma determinada falha de referência, em diferentes computadores e grupos de computadores. Apenas as combinações de eventos de falha de SO que apresentam maior consistência na amostra de trabalho são consideradas padrões de falhas de fato.

Rankings foram criados com o propósito de auxiliar a detecção dos **PF**s. Esses *rankings* foram compostos por **PFC**s ordenados pela sua consistência na amostra. Portanto, os primeiros **PFC**s dos *rankings* possuem maiores chances de serem considerados como padrões de falhas.

O primeiro estágio do protocolo organiza os registros de falha, seleciona as falhas de referência (**FR**s) e define os intervalos de pesquisa. O segundo estágio, representado pelo Algoritmo 1, localiza todos os **EFA**s e/ou **EFP**s para cada intervalo de pesquisa (**IP)**, os quais são usados nos estágios seguintes. O terceiro estágio seleciona os tipos/subtipos dos **EFA**s e/ou **EFP**s que ocorreram juntamente com as **FR**s em pelo menos três dos quatro grupos da amostra de trabalho, uma vez que é desejável encontrar padrões de falhas que se repetem sistematicamente, isto é, na maior quantidade de grupos possível. O quarto estágio, representado pelo Algoritmo 2, consiste na análise de **PFC**s com a configuração **PFC=EFA→FR**. O quinto e o sexto estágios correspondem à análise de **PFC**s com as outras duas configurações, **PFC=FR→EFP** e **PFC=EFA→FR→EFP**. Esses dois últimos estágios foram implementados seguindo a mesma lógica do quarto estágio. Por fim, o sétimo estágio consiste na aplicação dos critérios utilizados para a composição dos *rankings*. A Figura 4.1 exibe o fluxo de execução do protocolo.

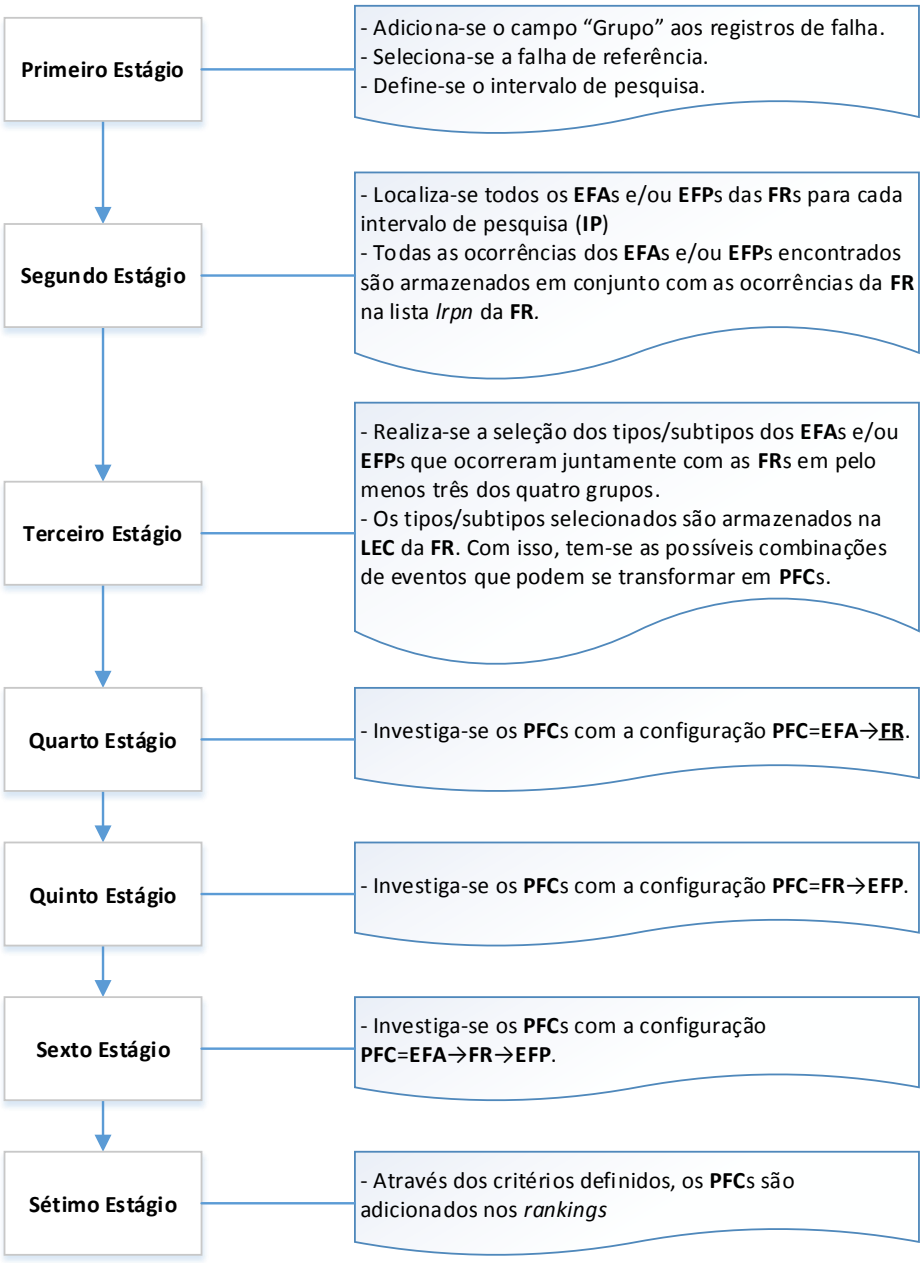


Figura 4.1. Fluxo de execução do protocolo.

4.3.1 Primeiro Estágio

Primeiramente, este estágio do protocolo organiza os registros de falha de SO e, posteriormente, as **FR**s e os **IP**s são definidos.

Todas as 7.007 falhas da amostra de trabalho foram armazenadas em um repositório de falhas. Este repositório contém todas as falhas de SO dos computadores presentes nos grupos da amostra de trabalho. Além dos dez campos originais (ver Seção 3.4), foi adicionado um novo campo aos registros de falha de SO (campo "Grupo").

Com o objetivo de identificar as falhas de referência (**FR**s), a ocorrência de todos os tipos/subtipos de falha de SO encontrados em cada grupo (G1, G2, G3 e G4) foi avaliada. Como a amostra de trabalho é composta por quatros grupos, foi definido que, neste estudo, as **FR**s são falhas de SO que devem estar presentes em

pelo menos três grupos. No total, 28 tipos/subtipos de falha de SO, que consistem de 6.351 eventos de falha, foram considerados como falhas de referência, dos quais 56,24% ocorreram em três grupos e 43,76% ocorreram nos quatro grupos. As Tabelas 4.2 e 4.3 exibem a distribuição dessas falhas entre os grupos.

Tabela 4.2. Número de ocorrências por tipo/subtipo de falha de SO observadas nos quatro grupos

Categoria	Tipo de falha de SO [subtipo]	G1	G2	G3	G4	Total
SO_{SVC}	SWU[Atualização do Windows]	23	613	5	523	1.164
SO_{APP}	explorer.exe	35	226	29	296	586
SO_{SVC}	SWU[Atualização do .Net Framework]	3	187	3	116	309
SO_{SVC}	SWU[Atualização do MS Office]	44	87	1	134	266
SO_{KNL}	Windows-StartupRepair	66	135	2	34	237
SO_{SVC}	msiexec.exe	31	3	3	23	60
SO_{SVC}	dllhost.exe	2	28	13	4	47
SO_{KNL}	SYSTEM_THREAD_ EXCEPTION_NOT HANDLED	1	30	7	3	41
SO_{SVC}	services.exe	1	16	5	19	41
SO_{APP}	mmc.exe	3	10	4	7	24
SO_{SVC}	taskhost.exe	1	1	1	1	4

Tabela 4.3. Número de ocorrências por tipo/subtipo de falha de SO observadas em três grupos

Categoria	Tipo de falha de SO [subtipo]	G1	G2	G3	G4	Total
SO_{SVC}	SWU[Atualização do Internet Explorer]	7	1.461		118	1.586
SO_{KNL}	VIDEO_TDR_ERROR	825	62		11	898
SO_{APP}	mscorsvw.exe		115	6	159	280
SO_{SVC}	SWU[Atualização do Visual Studio]		55	103	41	199
SO_{APP}	rundll32.exe		19	7	155	181
SO_{KNL}	DRIVER_POWER _STATE_FAILURE		22	23	78	123
SO_{SVC}	svchost.exe		28	24	53	105
SO_{SVC}	SWU[Atualização do MS C++]		16	5	19	40
SO_{SVC}	SWU[Atualização do XML]		16	1	16	33
SO_{KNL}	CRITICAL_ OBJECT_TERMINATION	18	4		6	28
SO_{KNL}	MEMORY _MANAGEMENT		22	1	2	25
SO_{KNL}	PAGE_FAULT_ IN_NONPAGED_AREA		7	1	10	18
SO_{KNL}	DRIVER_IRQL_ NOT_LESS_OR_EQUAL	1	11		5	17
SO_{SVC}	SWU[Atualização do MS Silverlight]		4	2	11	17
SO_{APP}	dwm.exe	4		1	4	9
SO_{SVC}	SWU[Atualização do Visio Viewer]	1	5		2	8
SO_{KNL}	CLOCK_WATCHDOG _TIMEOUT	1	3		1	5

Nota-se que todas as categorias de falhas (SO_{APP}, SO_{SVC} e SO_{KNL}) tiveram falhas consideradas como falhas de referência. Em alguns casos, apesar das falhas terem sido observadas em diversos grupos, a quantidade de ocorrências é baixa. Como exemplo, o tipo taskhost.exe, o qual é observado nos quatro grupos da amostra, porém, possui apenas uma ocorrência em cada um dos grupos (ver Tabela 4.2).

4.3.2 Segundo Estágio

Neste estágio, todos os eventos de falha anteriores (**EFA**s) e/ou todos os eventos de falha posteriores (**EFP**s), em relação às ocorrências das **FR**s e que estejam dentro dos **IP**s especificados, são coletados. O Algoritmo 1 mostra como os **EFA**s e/ou **EFP**s são encontrados para cada ocorrência das **FR**s.

Algoritmo 1 - Busca por EFAs e/ou EFPs.

```
01: fr: falha de referência.
02: rf: repositório de falhas.
03: TG: carimbo de hora (timestamp) do evento de falha .
04: ΔtA: intervalo de tempo entre o EFA e a FR.
05: ΔtP: intervalo  de tempo entre a FR e o EFP.
06: ip: intervalo de tempo para pesquisa.
07: lrpn: lista contendo a FR e seus EFAs e/ou EFPs.
08: for each position in rf
09:   if (rf[position]. ProductName is equal to fr) then
10:    offsetAntes = position - 1;
11:    ΔtA = rf[position].TG - rf[offsetAntes].TG;
12:    while (ΔtA is less or equal to ip)
13:       lrpn.Insert(rf[offsetAntes]);
14:       offsetAntes = offsetAntes - 1;
15:       ΔtA = rf[position].TG - rf[offsetAntes].TG;
16:    end while
17:    lrpn.Insert(rf[position]);
18:    offsetDepois = position + 1;
19:    ΔtP = rf[offsetDepois].TG - rf[position].TG;
20:    while (ΔtP is less or equal to ip)
21:       lrpn.Insert(rf[offsetDepois]);
22:       offsetDepois = offsetDepois + 1;
23:       ΔtP = rf[offsetDepois].TG – rf[position].TG;
24:    end while
25:  end if
26: end for
```

O campo *TimeGenerated* dos registros de falha do RAC foi utilizado para descobrir os **EFA**s e/ou **EFP**s. Este campo é representado no Algoritmo 1 pela variável *TG*. Com base em *TG*, o **ΔtA** e/ou **ΔtP** dos eventos de falha que ocorreram antes e/ou depois das **FR**s, respectivamente, são verificados para determinar quais desses eventos ocorreram dentro dos **IP**s especificados (linhas 12 e 20), com a finalidade de adicioná-los à lista *lrpn* (linhas 13 e 21).

A *lrpn* é uma lista que contém todas as ocorrências da **FR** sendo investigada, assim como os seus respectivos **EFA**s e/ou **EFP**s que estejam dentro de um determinado **IP**. Cada **FR** possui uma *lrpn* para cada **IP** considerado. Por exemplo, a **FR**=explorer.exe possui três listas *lrpn* (*lrpn*-explorer.exe-2h*, lrpn*-explorer.exe-4h *e lrpn*-explorer.exe-8h). A Figura 4.2 apresenta um exemplo de como as ocorrências de uma **FR**, neste caso **FR**=explorer.exe, e seus respectivos **EFA**s e/ou **EFP**s são armazenados em uma lista *lrpn* (*lrpn*-explorer.exe-2h). O intervalo de pesquisa considerado foi de 2 horas. Portanto, todos os eventos de falha que

ocorreram 2 horas antes e/ou depois das falhas de explorer.exe foram armazenados juntos com as ocorrências da **FR** na *lrpn*-explorer.exe-2h.

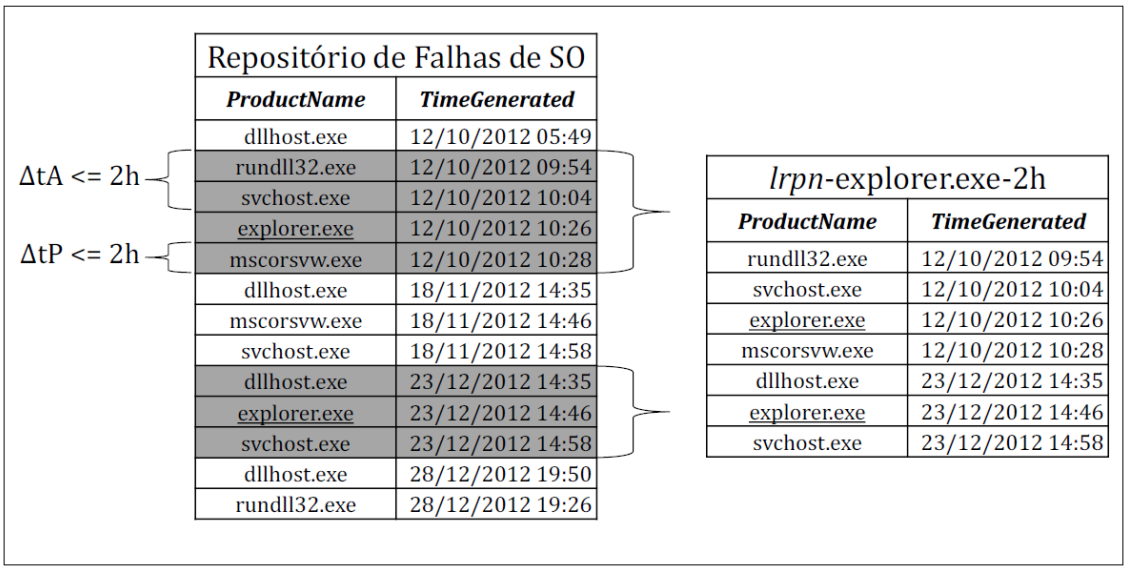


Figura 4.2. Exemplo de coleta das ocorrências da FR=explorer.exe e seus EFAs e EFPs.

4.3.3 Terceiro Estágio

Neste ponto, o protocolo analisa todos os **EFA**s e/ou **EFP**s presentes nas listas *lrpns*, a fim de selecionar os tipos/subtipos daqueles eventos de falha que ocorreram juntamente com uma determinada **FR** em pelo menos três dos quatro grupos da amostra de trabalho, uma vez que é desejado encontrar padrões de falhas consistentes.

Os tipos/subtipos dos eventos de falha selecionados são armazenados em uma lista de eventos candidatos (**LEC**). Portanto, cada **FR** possui uma **LEC**. A Tabela 4.4 apresenta um exemplo dos eventos de falha que foram adicionados na **LEC** para o caso **FR**=explorer.exe. Neste caso, apenas os tipos explorer.exe e dllhost.exe foram adicionados à **LEC**, visto que ocorreram em pelo menos três grupos juntamente com a **FR.**

Para o caso em que a **FR**=explorer.exe e a **LEC**-explorer.exe={explorer.exe, dllhost.exe}, existem oito possíveis padrões de falhas de SO candidatos, os quais estão listados na Tabela 4.5.

Tabela 4.4. Grupos em que os EFAs e/ou EFPs ocorreram juntamente com a FR=explorer.exe

Tipos/Subtipos dos EFAs e/ou EFPs	Grupos em que os EFAs e/ou EFPs estão presentes
explorer.exe	G1, G2, G3, G4
dllhost.exe	G1, G2, G4
~~spoolsv.exe~~	~~G2, G4~~

Tabela 4.5. Possíveis PFCs para FR=explorer.exe e LEC-explorer.exe={explorer.exe, dllhost.exe}

Configurações de PFCs	Combinações de Eventos (PFCs)
PFC =EFA→FR	explorer.exe → explorer.exe
	dllhost.exe → explorer.exe
PFC =FR→EFP	explorer.exe → explorer.exe
	explorer.exe → dllhost.exe
PFC=EFA→FR→EFP	explorer.exe → explorer.exe → explorer.exe
	explorer.exe → explorer.exe → dllhost.exe
	dllhost.exe → explorer.exe → explorer.exe
	dllhost.exe → explorer.exe → dllhost.exe

4.3.4 Quarto Estágio

O quarto estágio, sintetizado pelo Algoritmo 2, consiste em investigar **PFC**s a partir da análise de eventos de falha que ocorreram antes de uma falha de referência para um **IP** definido, ou seja, **PFC**=**EFA**→**FR**.

Primeiramente, definem-se as combinações de eventos que serão investigadas para a configuração **PFC**=**EFA**→**FR**. Utilizando o exemplo da etapa anterior, existem dois possíveis **PFC**s nesta etapa, **PFC**=explorer.exe→explorer.exe e **PFC**=dllhost.exe→explorer.exe (ver Tabela 4.5).

Em seguida, por meio da análise do tipo/subtipo dos registros de falha de SO, o protocolo identifica as ocorrências da **FR** investigada (linha 12) em uma de suas *lrpns*. Ao invés de verificar os **EFA**s, primeiramente verificam-se os eventos de falha que ocorreram após a ocorrência da **FR** investigada, isto é, **EFP**s que estejam dentro do **IP** definido. Comparam-se os tipos/subtipos dos **EFP**s com os tipos/subtipos presentes na **LEC** (linha 16), e caso sejam distintos, analisam-se os **EFA**s. Dessa forma, verifica-se a existência de pelo menos um evento de falha com o mesmo tipo/subtipo de falha do **EFA** referente ao **PFC** sendo pesquisado (linha 28) e, caso seja encontrado, buscam-se mais **EFA**s com este mesmo tipo/subtipo e que estejam dentro do intervalo de pesquisa definido. Ao final, os **EFA**s encontrados são armazenados junto com a ocorrência da **FR** em uma lista *lPFC* (linhas 29 e 35). Essa lista contém todas as ocorrências de um **PFC** para um determinado **IP**. Por exemplo, *lPFC*-(dllhost.exe-explorer.exe)-2h contém todas as ocorrências do **PFC**=dllhost.exe→explorer.exe para o IP=2 horas, assim como a *lPFC*-(dllhost.exe-explorer.exe-dllhost.exe)-8h possui todas as ocorrências do **PFC**=dllhost.exe→explorer.exe→ dllhost.exe para o IP=8 horas.

Algoritmo 2 Investigação do padrão PFC=EFA→FR

```
01: fr: falha de referência
02: efa:evento de falha anterior
03: lrpn: lista contendo a FR e seus EFAs e/ou EFPs
04: lec: lista dos eventos candidatos
05: TG: carimbo de hora (timestamp) do evento de falha
06: ΔtA: interval de tempo entre o EFA e a FR
07: ΔtP: interval de tempo entre a FR e o EFP
08: ip: intervalo de tempo para pesquisa
09: lPFC: lista contendo as ocorrências do PFC investigado
10: lPFCTamInicial: Tamanho inicial da lista lPFC
11: for each position in lrpn
12:   if (lrpn[position]. ProductName is equal to fr) then
13:     offsetDepois = position + 1;
14:     ΔtP = lrpn[offsetDepois].TG - lrpn[position].TG;
15:     while (ΔtP is less or equal to ip)
16:       if (lrpn[offsetDepois].ProductName is equal to any Tipo/Subtipo in lec) then
17:         flag = 1;
18:         break;
19:       else
20:         offsetDepois = offsetDepois + 1;
21:         ΔtP = lrpn[position].TG - lrpn[offsetDepois].TG;
22:       end if
23:     end while
24:     if (flag is different to 0) then
25:       offsetAntes = position - 1;
26:       ΔtA = lrpn[position].TG - lrpn[offsetAntes].TG;
27:       while (ΔtA is less or equal to ip)
28:         if (lrpn[offsetAntes]. ProductName is equal to efa) then
29:           lPFC.Insert(lrpn.[ offsetAntes]);
30:         end if
31:         offsetAntes = offsetAntes – 1;
32:         ΔtA = lrpn[position].TG - lrpn[offsetAntes].TG;
33:       end while
34:       if (lPFC.Size is larger than lPFCTamInicial )
35:       lPFC.Insert(lrpn.[ position]);
36:       lPFCTamInicial = lPFC.Size;
37:     end if
38:   end if
39: end for
```

A Figura 4.3 mostra um exemplo da execução do quarto estágio do protocolo (ver Algoritmo 2). Neste exemplo, a combinação de eventos investigada é representada por **FR**=explorer.exe e **EFA**=dllhost.exe, sendo a lista de eventos candidatos denominada de **LEC**-explorer.exe={explorer.exe, dllhost.exe} e o intervalo de pesquisa é de 2 horas.

Primeiramente, o protocolo identifica uma ocorrência da **FR** investigada. Por exemplo, a falha com o tipo explorer.exe que ocorreu às 13:10 (ver Figura 4.3). Em seguida, verificam-se os eventos de falha que ocorreram após a ocorrência da **FR**=explorer.exe$^{13:10}$ e que estejam dentro do **IP** considerado. Neste caso, existe apenas o **EFP**=dllhost.exe$^{13:11}$. Analisando a **LEC**-explorer.exe, verifica-se que o tipo dllhost.exe pertence a esta lista. Portanto, essa ocorrência não representa a combinações de eventos investigada (**PFC**=dllhost.exe→explorer.exe), pois o algoritmo elimina o caso em que os **EFP**s possuem os mesmos tipos encontrados na **LEC**. Esses casos são eliminados neste estágio, pois serão analisados no sexto estágio do protocolo. Por exemplo, essa primeira ocorrência da **FR** às 13:10 (ver

Figura 4.3), forma a combinação de eventos representada por **PFC**=dllhost.exe→explorer.exe→dllhost.exe, a qual é uma possível combinação eventos para a configuração **PFC=EFA→FR→EFP** (ver Tabela 4.5), que é analisada no sexto estágio do protocolo. Portanto, sempre que a configuração **PFC=EFA→FR→EFP** ocorre, as outras duas possíveis configurações, isto é, **PFC=EFA→FR** e **PFC= FR→EFP**, não ocorrem.

Quando o protocolo identifica uma segunda ocorrência da **FR** investigada (**FR**=explorer.exe$^{18:26}$), verifica-se que esta representa uma ocorrência do **PFC** investigado. Isso ocorreu, pois os **EFP**s em relação à ocorrência da **FR** não são dos mesmos tipos/subtipos presentes na **LEC**-explorer.exe, ou seja, nenhum possível **PFC** com a configuração **PFC=EFA→FR→EFP** foi encontrado para essa segunda ocorrência da **FR.** Além disso, encontrou-se um evento de falha que possui o mesmo tipo/subtipo de falha do **EFA** referente ao **PFC** sendo pesquisado (**PFC**=dllhost.exe→explorer.exe). Na Figura 4.3, a combinação de eventos que compõe o **PFC** encontrado (**PFC**=dllhost.exe$^{18:20}$→explorer.exe$^{18:26}$) é destacada pelo símbolo (*).

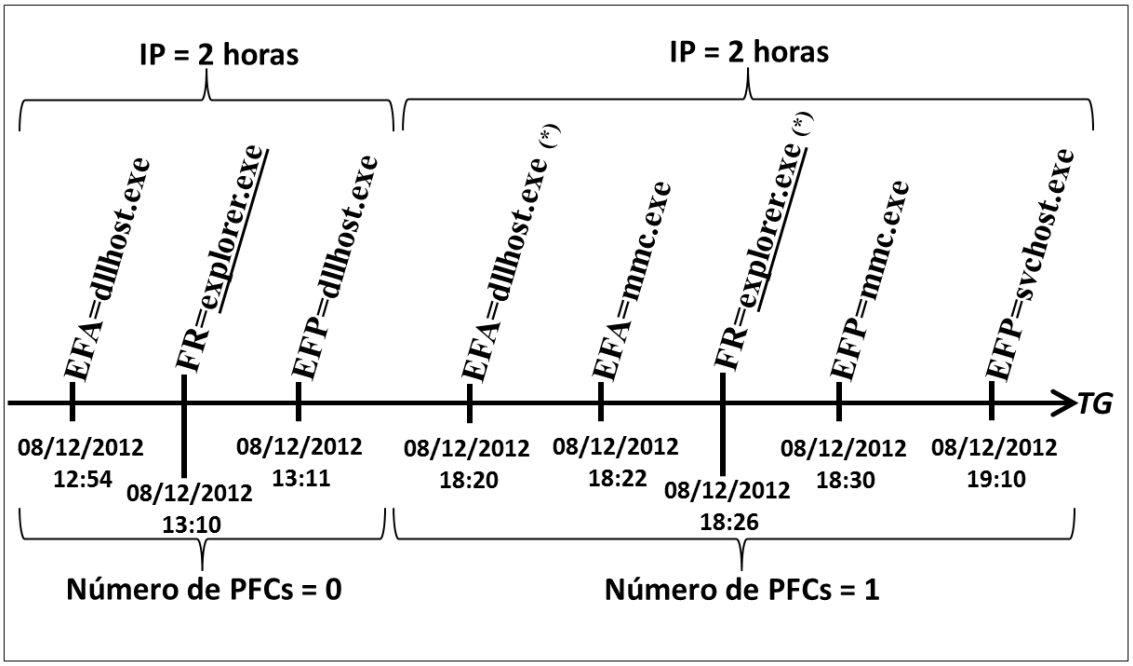


Figura 4.3. Análise do PFC=dllhost.exe→explorer.exe.

4.3.5 Quinto Estágio

No quinto estágio do protocolo, consiste em investigar **PFC**s a partir da análise de eventos de falha que ocorreram depois de uma **FR** para um **IP** definido, **PFC=FR→EFP**. Neste caso, após a identificação de uma ocorrência da **FR** investigada em uma de suas *lrpns*, verificam-se primeiramente os eventos de falha que ocorreram antes da ocorrência da **FR** investigada, isto é, **EFA**s que estejam dentro do **IP** investigado. Se não forem encontrados registros de falha com os tipos/subtipos presentes na **LEC** antes da ocorrência da **FR**, ou seja, se nenhum possível **PFC** com a configuração **PFC=EFA→FR→EFP** foi encontrado, verifica-se a existência de pelo menos um **EFP** que condiz com o **PFC** pesquisado. Caso seja encontrado, verifica-se a existência de outros **EFP**s do mesmo tipo/subtipo ocorrendo depois da mesma ocorrência da **FR** e que estejam dentro do intervalo de

pesquisa definido. Ao final, os **EFP**s encontrados são armazenados junto com a ocorrência da **FR** em uma lista *lPFC*.

4.3.6 Sexto Estágio

Neste estágio o protocolo investiga **PFC**s com a configuração **PFC=EFA→FR→EFP**. Após a identificação de uma ocorrência da **FR** investigada em uma de suas *lrpns*, verifica-se a existência de pelo menos uma ocorrência do **EFA** e, necessariamente, de pelo menos uma ocorrência do **EFP** que corresponde ao **PFC** investigado. Caso sejam encontradas, outros **EFA**s e **EFP**s com os mesmos tipos/subtipos e que estejam dentro do intervalo de pesquisa definido são investigados para a mesma ocorrência da **FR**. Ao final, os **EFA**s e **EFP**s encontrados são armazenados junto com a ocorrência da **FR** em uma lista *lPFC*.

4.3.7 Sétimo Estágio

No sétimo estágio, as listas *lPFC* são analisadas. Primeiramente, por meio do campo (Grupo) adicionado aos registros de falha (ver Seção 4.3.1), calcula-se a quantidade de grupos com as ocorrências dos **PFC**s, obtidas nos estágios anteriores, para um determinado **IP**. Por meio do campo *ComputerName*, calcula-se a quantidade de computadores com as ocorrências dos **PFC**s. Posteriormente, a quantidade de **EFA**s e/ou **EFP**s é contabilizada para identificar a quantidade de ocorrências dos **PFC**s. Por fim, calcula-se o desvio padrão dos **ΔtA**s e/ou **ΔtP**s das ocorrências do **PFC**. Por meio da análise dos valores obtidos os **PFC**s são posicionados nos *rankings*.

4.4 Exemplo de Uso

Nesta seção é apresentado um exemplo do funcionamento do protocolo proposto para detecção e caracterização de padrões de falhas de SO, o qual foi descrito na seção anterior. Este exemplo não faz parte dos resultados obtidos neste estudo, sendo criado apenas para fins didáticos. A Tabela 4.6 exibe um exemplo do repositório de falhas de SO. Como já destacado na Seção 4.3.1, o campo "Grupo" foi adicionado com o propósito de identificar de qual dos quatro grupos da amostra de trabalho os registros são provenientes.

Tabela 4.6. Exemplo do repositório de falhas

Grupo	*ComputerName*	*ProductName*	*TimeGenerated*
G1	Comp1	dllhost.exe	12/10/2012 05:49
G1	Comp1	explorer.exe	12/10/2012 10:02
G1	Comp1	svchost.exe	12/10/2012 10:04
G1	Comp1	dllhost.exe	12/10/2012 10:26
G2	Comp5	mscorsvw.exe	12/10/2012 10:28
G2	Comp5	dllhost.exe	18/11/2012 14:35
G2	Comp5	mscorsvw.exe	18/11/2012 14:46
G2	Comp5	svchost.exe	18/11/2012 14:58
...	...	...	...

No estágio inicial, definem-se a falha de referência e o intervalo de pesquisa. Neste exemplo, ambos são representados por **FR**=svchost.exe e **IP**=2 horas. Posteriormente, todos os **EFA**s e/ou todos os **EFP**s que ocorreram 2 horas antes e/ou depois, em relação às ocorrências da **FR**=svchost.exe, são coletados e adicionados na *lrpn*-svchost.exe-2h. Todos os **EFA**s e/ou todos os **EFP**s que ocorreram nos outros **IP**s (4 e 8 horas) também são coletados. Isso representa a segunda etapa do protocolo. A Figura 4.4 apresenta um exemplo de como as ocorrências da **FR**=svchost.exe e seus respectivos **EFA**s e/ou **EFP**s são armazenados em uma lista *lrpn* (*lrpn*-svchost.exe.exe-2h) para o **IP**=2 horas.

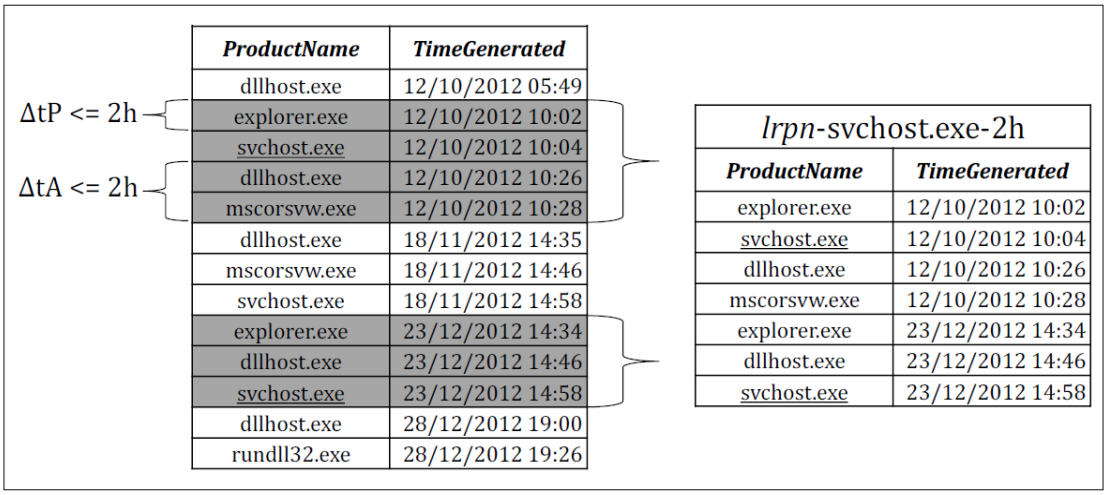


Figura 4.4. Exemplo de coleta das ocorrências da FR=svchost.exe e seus respectivos EFAs e/ou EFPs.

Após a captura de todos os eventos de falha com o tipo svchost.exe e seus respectivos **EFA**s e/ou **EFP**s, para o **IP** de 2 horas, realiza-se a seleção dos tipos/subtipos dos **EFA**s e/ou **EFP**s que ocorreram juntamente com as falhas de svchost.exe em pelo menos três dos quatro grupos. Como apresentado na Figura 4.5, todas as *lrpns* são analisadas. Os tipos/subtipos selecionados são armazenados na **LEC** da **FR** sendo investigada. Apenas eventos de falha com o tipo explorer.exe foram encontrados próximos aos eventos de falha do svchost.exe. Dessa forma, somente o tipo explorer.exe foi adicionado à **LEC**-svchost.exe. Portanto, existem três combinações de eventos (**PFC**s) para as três possíveis configurações de **PFC**s. Essas combinações estão apresentadas na Tabela 4.7.

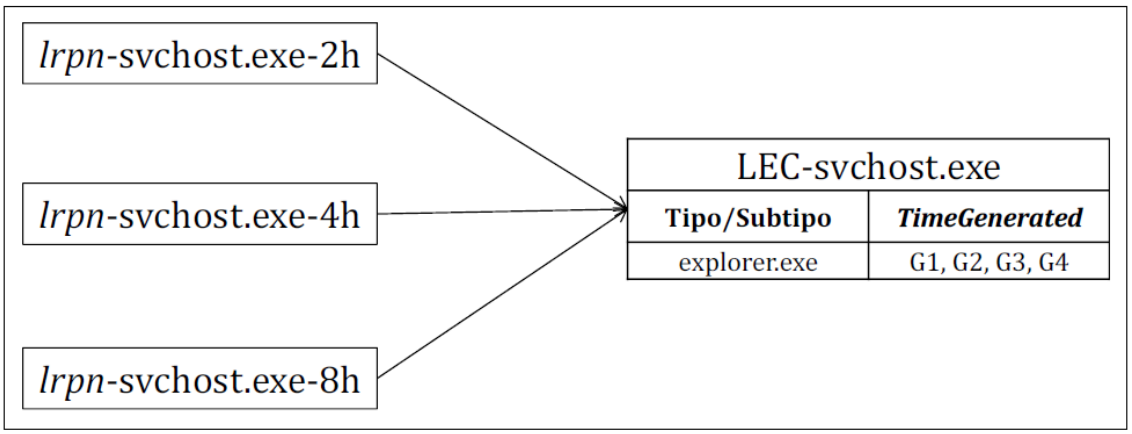


Figura 4.5. Seleção dos tipos/subtipos dos EFAs e/ou EFPs que ocorreram juntamente com a FR=explorer.exe em pelo menos 3 grupos.

Tabela 4.7. Possíveis PFCs para FR=svchost.exe e LEC-svchost.exe={explorer.exe}

Configurações de PFCs	Combinações de Eventos (PFCs)
PFC=EFA→FR	explorer.exe→svchost.exe
PFC=FR→EFP	svchost.exe→explorer.exe
PFC=EFA→FR→EFP	explorer.exe→svchost.exe→explorer.exe

Já no quarto estágio do protocolo, inicialmente definem-se as combinações de eventos que serão investigadas para a configuração **PFC=EFA→FR**. Para o exemplo apresentado nesta seção, existe apenas a combinação de eventos representada por **PFC**=explorer.exe→svchost.exe (ver Tabela 4.7). A Tabela 4.8 apresenta o conteúdo de uma *lrpn*. As linhas sombreadas destacam as ocorrências do **PFC** investigado em diferentes grupos e computadores. Neste ponto, as combinações encontradas na *lrpn* são inseridas em uma nova lista, a *lPFC*-(explorer.exe-svchost.exe)-2h, apresentada na Tabela 4.9.

Tabela 4.8. Resumo do conteúdo da *lrpn*-svchost.exe-2h

ComputerName	***ProductName***	***TimeGenerated***	Grupo
Comp1	explorer.exe	12/10/2012 10:02	G1
Comp1	svchost.exe	12/10/2012 10:04	
Comp1	dllhost.exe	12/10/2012 10:26	
Comp1	mscorsvw.exe	12/10/2012 10:28	
Comp14	explorer.exe	23/12/2012 14:34	G1
Comp14	dllhost.exe	23/12/2012 14:46	
Comp14	svchost.exe	23/12/2012 14:58	
Comp3	mscorsvw.exe	03/11/2010 10:28	G2
Comp3	explorer.exe	04/11/2010 18:24	
Comp3	explorer.exe	04/11/2010 18:36	
Comp3	svchost.exe	04/11/2010 18:48	
Comp23	mmc.exe	23/12/2012 14:35	G3
Comp23	explorer.exe	23/12/2012 14:46	
Comp23	svchost.exe	23/12/2012 14:58	
Comp4	svchost.exe	28/12/2012 19:00	G4
Comp4	explorer.exe	28/12/2012 19:26	

Tabela 4.9. Resumo do conteúdo da *lPFC*-(explorer.exe-svchost.exe)-2h

ComputerName	*ProductName*	*TimeGenerated*	Grupo
Comp1	explorer.exe	12/10/2012 10:02	G1
Comp1	svchost.exe	12/10/2012 10:04	
Comp14	explorer.exe	23/12/2012 14:34	G1
Comp14	svchost.exe	23/12/2012 14:58	
Comp3	explorer.exe	04/11/2010 18:24	G2
Comp3	explorer.exe	04/11/2010 18:36	
Comp3	svchost.exe	04/11/2010 18:48	
Comp23	explorer.exe	23/12/2012 14:46	G3
Comp23	svchost.exe	23/12/2012 14:58	

Ao final, por meio da análise das ocorrências do **PFC** na *lPFC* (Tabela 4.9), calcula-se a quantidade de grupos e computadores com as ocorrências do **PFC** investigado. Computa-se a quantidade de ocorrências do **PFC** e, por fim, calcula-se o desvio padrão dos **ΔtA**s e/ou **ΔtP**s em horas. Dessa forma, os resultados calculados para o **PFC**=explorer.exe→svchost.exe, para o **IP**=2 horas são analisados. Por meio de uma análise comparativa contra os outros **PFC**s com o mesmo **IP** (2 horas), o **PFC** em questão é posicionado no *ranking* **PFC=EFA→FR** (2h). A Tabela 4.10 apresenta um exemplo fictício desse *ranking*.

Tabela 4.10. Exemplo do *ranking* PFC=EFA→FR (2h)

IP	Rank	EFA	FR	Nº de Grupos	Nº de Computadores com o PFC	Nº de Ocorrências do PFC	DP
2 horas	**1**	explorer.exe	explorer.exe	4	6	8	0,11
	2	dllhost.exe	explorer.exe	4	2	9	0,18
	3	explorer.exe	svchost.exe	3	3	5	0,16
	4	mmc.exe	dllhost.exe	3	2	4	0,25
	5	...	...		...	...	...

No primeiro critério, quantidade de grupos com o **PFC**, somente os **PFC**s que ocorreram em pelo menos três grupos, foram adicionados ao *ranking*. Na Tabela 4.10, a linha sombreada, na terceira posição do *ranking*, destaca o **PFC** analisado nesta seção (**PFC**=explorer.exe→svchost.exe). Nota-se, que apesar dele estar presente em mais computadores que o **PFC** da segunda posição (**PFC**=dllhost.exe→explorer.exe), ele ocorreu em menos grupos.

4.5 Detecção e Caracterização de Padrões de Causas de Falhas

Com o propósito de entender a relação entre a causa de uma falha e a causa de outras falhas em seu entorno, o protocolo apresentado na seção anterior foi adaptado para realizar a detecção e caracterização de padrões das causas das falhas de SO observadas na amostra de trabalho. Esta abordagem foi aplicada para

identificar as características, isto é, as categorias, tipos/subtipos e causas de falha (*Error Codes*) que ocorreram imediatamente antes e/ou imediatamente depois das falhas que apresentaram maior consistência na amostra.

4.5.1 As Causas Mais Frequentes de Falhas do SWU

A categoria Serviço de SO (SO_{SVC}) possui o maior número de ocorrências de falhas de SO na amostra de trabalho (60,63% das falhas de SO). As falhas do serviço Windows *Update* (SWU), pertencentes à categoria SOsvc, representam a maior parte dessas falhas (87,97% do total de falhas de SO_{SVC}). Portanto, as causas dessas falhas foram analisadas detalhadamente com o propósito de identificar o motivo da alta frequência de falhas pertencentes ao tipo SWU.

Como apresentado na Seção 3.6.1, quatro subtipos (AIE, AW7, ANF e AMO) foram responsáveis por 88,98% de todas as falhas relacionadas com o SWU. As causas de falha desses subtipos foram identificadas por meio dos *Error Codes* presentes no campo *Message* dos seus registros de falha.

Inicialmente, analisaram-se todos os eventos de falha de cada um dos quatro subtipos de SWU separadamente, com a intenção de identificar as três causas de falha (*Error Codes*) mais frequentes por subtipo. Além disso, foram investigadas as três causas de falha mais frequentes considerando todas as falhas do SWU, isto é, os 23 subtipos (ver Seção 3.6.1) relacionados com o tipo serviço Windows *Update* (SWU). A Tabela 4.11 apresenta os resultados dessa análise. O termo "Total", nas duas primeiras colunas, representa a quantidade total de *Error Codes* e a quantidade total de eventos de falha, respectivamente, do subtipo analisado.

O *Error Code* 0x80070643 foi o mais recorrente em três dos quatro subtipos analisados. Este código indica uma falha durante o processo de instalação de uma atualização de *software*, cuja causa mais comum é o mau funcionamento do .Net Framework instalado no sistema [Microsoft 2016j], [Microsoft 2016k]. Apesar deste *Error Code* não estar presente na Tabela 4.11, para o subtipo Atualização do .Net Framework, ele foi observado em 23 eventos de falha para esse subtipo. Após a análise dos eventos de falha de todos os 23 subtipos do SWU, verificou-se que este *Error Code* representa a maior causa de falhas do SWU, estando presente em 61,21% dos eventos de falha de atualização de *software* da amostra de trabalho.

Outro *Error Code* observado em três dos quatro subtipos analisados foi o 0x800706ba. Apesar deste código não estar presente na Tabela 4.11 para o subtipo AIE, ele foi observado em 20 eventos de falha desse subtipo. Este *Error Code* indica uma falha no SWU causada pela indisponibilidade do servidor RPC (*Remote Procedure Call*), o qual é amplamente utilizado pelo SWU e outros serviços do Win7 [Microsoft 2016l]. Este *Error Code* esta presente em 11,99% das falhas do SWU, sendo a segunda maior causa de falhas de atualização de *software* observada na amostra de trabalho.

O *Error Code* 0x800b0100 aparece em falhas de dois subtipos (Atualizado do Windows e .Net Framework). Este código indica que os arquivos necessários para o SWU atualizar um determinado *software* estão faltando ou estão corrompidos [Microsoft 2016m]. Esta é a terceira maior causa de falhas do SWU, presente em 7,22% dos eventos de falha.

Os outros códigos listados na Tabela 4.11 foram os mais frequentes em apenas um dos quatro subtipos relevantes. No entanto, eles também foram observados em percentuais inferiores nos outros subtipos.

Tabela 4.11. As três causas de falha (*Error Codes*) mais frequentes em falhas de atualização de software

Atualização do Internet Explorer (AIE)		
***Error Codes* (Total = 18)**	**Nº de Eventos (Total = 1.586)**	**% de Eventos**
0x80070643	1.450	91,42%
0x80242016	60	3,78%
0x80073712	36	2,27%
Atualização do Windows (AW7)		
***Error Codes* (Total = 24)**	**Nº de Eventos (Total = 1.164)**	**% de Eventos**
0x80070643	389	33,42%
0x800706ba	302	25,95%
0x800b0100	112	9,62%
Atualização do .Net Framework (ANF)		
***Error Codes* (Total = 17)**	**Nº de Eventos (Total = 309)**	**% de Eventos**
0x800b0100	154	49,84%
0x800706ba	46	14,89%
0xc8000710	24	7,77%
Atualização do MS Office (AMO)		
***Error Codes* (Total = 11)**	**Nº de Eventos (Total = 266)**	**% de Eventos**
0x80070643	97	36,47%
0x80070652	66	24,81%
0x800706ba	65	24,44%
Geral - 23 subtipos (SWU)		
***Error Codes* (Total = 39)**	**Nº de Eventos (Total = 3.737)**	**% de Eventos**
0x80070643	2.287	61,20%
0x800706ba	448	11,99%
0x800b0100	266	7,12%

O *Error Code* 0xc8000710 prevaleceu em eventos de falha de ANF e também foi observado nos outros subtipos (1 evento de falha em AIE, 25 eventos em AMO e 54 eventos em AW7). De acordo com [Microsoft 2016n], este código indica que a falha ocorreu devido à indisponibilidade de espaço em disco durante as atualizações.

Os *Error Codes* 0x80242016 and 0x80073712 foram observados principalmente em falhas de AIE. O primeiro também foi observado em falhas de AW7 (31 eventos) e ANF (9 eventos), enquanto o segundo também foi observado em falhas de AW7 (24 eventos).

O *Error Code* 0x80242016 indica uma falha devido a uma perda de conexão entre o serviço Windows *Update* e seus servidores [Microsoft 2016o]. O *Error Code*

0x80073712 indica uma falha durante o processo de instalação de uma atualização, causada pela corrupção do CBS (*Component-based servicing*) do Win7 [Microsoft 2016p]. O CBS é a infraestrutura responsável por instalar as atualizações [Technet 2016], [Lee *et al.* 2011].

O *Error Code* 0x80070652 foi observado em falhas de AMO. Este código indica que, quando a instalação foi iniciada, havia outra instalação em andamento e ambas não podem ser executadas simultaneamente [Microsoft 2016q].

4.5.2 Distribuição das Causas de Falha do SWU

Além da análise das causas mais frequentes de falhas do SWU (ver Tabela 4.11), investigou-se também a distribuição dessas causas na amostra de trabalho em relação aos diferentes grupos e computadores. Dessa forma, a distribuição dos eventos de falha que possuem os *Error Codes* mais frequentes, listados na Tabela 4.11, foi examinada. Os resultados são apresentados na Tabela 4.12.

Tabela 4.12. As três causas de falha (*Error Codes*) mais frequentes por grupo e número de computadores

Atualização do Internet Explorer (AIE)					
Error Codes	**G1**	**G2**	**G3**	**G4**	**Total**
0x80070643	-	1.386 (3)	-	64 (13)	1.450 (16)
0x80242016	6 (3)	28 (14)	-	26 (19)	60 (36)
0x80073712	-	35 (1)	-	1 (1)	36 (2)
Atualização do Windows (AW7)					
Error Codes	**G1**	**G2**	**G3**	**G4**	**Total**
0x80070643	-	347 (3)	-	42 (3)	389 (6)
0x800706ba	1 (1)	128 (4)	-	173 (3)	302 (8)
0x800b0100	-	64 (4)	-	48 (1)	112 (5)
Atualização do .Net Framework (ANF)					
Error Codes	**G1**	**G2**	**G3**	**G4**	**Total**
0x800b0100	-	154 (2)	-	-	154 (2)
0x800706ba	2 (2)	13 (3)	-	31 (3)	46 (8)
0xc8000710	-	-	-	24 (1)	24 (1)
Atualização do MS Office (AMO)					
Error Codes	**G1**	**G2**	**G3**	**G4**	**Total**
0x80070643	32 (32)	4 (4)	-	61 (7)	97 (43)
0x80070652	-	60 (15)	-	6 (5)	66 (20)
0x800706ba	11 (1)	21 (2)	-	33 (2)	65 (5)
Geral - 23 subtipos (SWU)					
Error Codes	**G1**	**G2**	**G3**	**G4**	**Total**
0x80070643	32 (32)	1.839 (37)	111 (14)	305 (49)	2.287 (132)
0x800706ba	15 (2)	179 (5)	-	254 (3)	448 (10)
0x800b0100	-	218 (6)	-	48 (1)	266 (7)

Os valores entre parênteses da Tabela 4.12 indicam o número de computadores no grupo em que uma determinada falha de SO foi causada pelos respectivos *Error Codes*.

Pode-se observar que as causas de falha mais frequentes na amostra de trabalho ocorreram nos grupos G2 e G4. Estes grupos possuem o maior percentual de falhas de SO na amostra, contendo 49,17% e 30,91% dos eventos de falha, respectivamente. Com relação às ocorrências da mesma causa de falha em diferentes computadores e grupos, o *Error Code* 0x80070643 é predominante, considerando os 23 subtipos de falha do SWU.

Na maioria dos casos, os *Error Codes* estão distribuídos em falhas que ocorreram em mais de um computador por grupo. No entanto, há casos em que o número total de falhas com o mesmo *Error Code* foi concentrado em um único computador de um grupo (ex. *Error Code* 0x800b0100 no grupo G4 para o subtipo AW7). Além disso, houve um caso (*Error Code* 0x80070643 no grupo G1 para o subtipo AMO), em que cada evento de falha com o mesmo *Error Code* foi observado apenas uma vez por computador no mesmo grupo.

Diante dessas diferentes distribuições observadas para cada causa de falha analisada e, com o propósito de compreender a relação entre a causa de uma determinada falha e a causa de outras falhas em seu entorno, uma extensão do protocolo para detectar padrões entre as causas de falha foi elaborada.

Essa abordagem investiga as falhas que ocorreram imediatamente antes e/ou imediatamente depois de cada um dos eventos de falha do SWU causados pelos *Error Codes* mais frequentes na amostra de trabalho. Neste contexto, as falhas de referência são conjunções, isto é, *Error Codes* e subtipos do SWU. Por exemplo, *Error Code*=0x80242016 e subtipo de falha do SWU=AIE. Portanto cada uma das quinze conjunções da Tabela 4.11 foi analisada como falha de referência, com o propósito de identificar os eventos que ocorreram imediatamente antes e/ou imediatamente depois dessas conjunções.

A Tabela 4.13 mostra os percentuais de eventos de falha pertencentes a cada uma das categorias de falha de SO (SO_{KNL}, SO_{APP}, and SO_{SVC}) e ao tipo serviço Windows *Update* (SWU), os quais ocorreram imediatamente antes e/ou imediatamente depois dos eventos de falha de cada uma das conjunções de referência.

Para todas as conjunções analisadas, os percentuais da coluna SO_{SVC} prevaleceram quando comparados com os valores das colunas SO_{KNL} e SO_{SVC} (ver Tabela 4.13). Isso demonstra que a maioria dos eventos de falha imediatamente antes e/ou imediatamente depois das conjunções analisadas pertencem à categoria SO_{SVC}.

Observou-se também que os valores da coluna SO_{SVC} são próximos aos valores da coluna SWU. Portanto, é possível concluir que existem fortes evidências de correlação entre os eventos de falha da categoria SO_{SVC}, especialmente com relação a falhas de atualização de *software*. Nenhuma evidência substancial de correlação entre as conjunções analisadas e as falhas de outras categorias foi encontrada (SO_{APP} e *Kernel*).

Tabela 4.13. Eventos de falha imediatamente antes e/ou imediatamente depois das conjunções (*Error Code* e subtipo do SWU) de referência

Atualização do Internet Explorer (AIE)				
Error Codes	**SO_{KNL}**	**SO_{APP}**	**SO_{SVC}**	**SWU**
0x80070643	0,52%	0,66%	98,82%	98,61%
0x80242016	6,19%	8,25%	85,57%	81,44%
0x80073712	0,0%	0,0%	100%	100%
Atualização do Windows (AW7)				
Error Codes	**SO_{KNL}**	**SO_{APP}**	**SO_{SVC}**	**SWU**
0x80070643	0,0%	0,78%	99,22%	99,22%
0x800706ba	0,17%	0,33%	99,50%	99,50%
0x800b0100	0,92%	0,46%	98,62%	98,62%
Atualização do .Net Framework (ANF)				
Error Codes	**SO_{KNL}**	**SO_{APP}**	**SO_{SVC}**	**SWU**
0x800b0100	0,0%	0,66%	99,34%	99,02%
0x800706ba	1,09%	2,17%	96,74%	96,74%
0xc8000710	0,0%	0,0%	100%	100%
Atualização do MS Office (AMO)				
Error Codes	**SO_{KNL}**	**SO_{APP}**	**SO_{SVC}**	**SWU**
0x80070643	18,59%	1,92%	79,49%	78,21%
0x80070652	6,72%	5,04%	88,24%	84,87%
0x800706ba	0,77%	0,77%	98,46%	98,46%
Geral - 23 subtipos (SWU)				
Error Codes	**SO_{KNL}**	**SO_{APP}**	**SO_{SVC}**	**SWU**
0x80070643	1,88%	1,48%	96,64%	95,79%
0x800706ba	0,33%	0,56%	99,11%	99,11%
0x800b0100	0,38%	0,57%	99,04%	98,85%

4.5.3 Padrões Entre as Causas de Falha do SWU

Como a maior parte dos eventos de falha imediatamente antes e/ou imediatamente depois das conjunções (*Error Codes* e subtipos do SWU) mais frequentes da amostra são de falhas de atualização de *software*, três cenários foram criados para detectar e caracterizar padrões entre as causas de falha do SWU. Portanto, os cenários foram utilizados com o propósito de compreender a relação entre a causa de uma determinada falha de atualização de software e a causa de outras falhas em seu entorno. Os três cenários criados foram:

1. As falhas imediatamente antes e/ou imediatamente depois devem possuir o mesmo *Error Code*, mas um subtipo diferente da conjunção de referência.

2. As falhas imediatamente antes e/ou imediatamente depois devem possuir o mesmo *Error Code* e o mesmo subtipo da conjunção de referência.

3. As falhas imediatamente antes e/ou imediatamente depois devem possuir o mesmo *Error Code*, subtipo e *componente* de atualização (*Update Component*) da conjunção de referência.

O primeiro cenário indica uma causa comum de falhas do SWU de diferentes subtipos. O segundo indica uma causa comum de falhas do SWU do mesmo subtipo. O terceiro cenário é um caso especial do segundo cenário, que indica uma causa comum de falhas do SWU do mesmo subtipo e para o mesmo *componente* de atualização.

O primeiro e o segundo cenários não consideram o *componente* de atualização. Este *componente* representa o elemento de *software* sendo atualizado no momento em que o SWU falhou. Assim, por meio do terceiro cenário, pode-se concluir que a atualização de um mesmo *componente* de *software* falhou múltiplas vezes.

A Tabela 4.14 mostra exemplos de cada um dos cenários, os quais foram extraídos da amostra de trabalho. As linhas sombreadas representam as conjunções analisadas (conjunções de referência). As linhas que aparecem acima e abaixo das linhas sombreadas representam as falhas imediatamente antes e/ou imediatamente depois, respectivamente.

Tabela 4.14. Exemplo dos três cenários analisados

Cenários	***Error Codes***	**Subtipo**	***ProductName***
1º	0x80242016	AIE	*Update* for Internet Explorer 8 Compatibility View List for Windows 7 for x64-based Systems (KB2598845)
	0x80242016	AW7	*Update* for Windows 7 for x64-based Systems (KB2703157)
	0x80242016	AIE	Security *Update* for Internet Explorer 8 for Windows 7 (KB2544521)
2º	0x800706ba	AW7	*Update* for Windows 7 for x64-based Systems (KB2541014)
	0x800706ba	AW7	Security *Update* for Windows 7 for x64-based Systems (KB2509553)
	0x800706ba	AW7	*Update* for Windows 7 for x64-based Systems (KB2488113)
3º	0x80070643	AW7	Windows 7 Service Pack 1 for x64-based Systems (KB976932)
	0x80070643	AW7	Windows 7 Service Pack 1 for x64-based Systems (KB976932)
	0x80070643	AW7	Windows 7 Service Pack 1 for x64-based Systems (KB976932)

Por meio do campo *ProductName* dos registros de falha foi possível identificar o *componente* de *software* que estava sendo atualizado quando a falha ocorreu, por meio do *Knowledge Base Identifiers* (KBs) [Thomas 2012], presentes neste campo. Os KBs também foram observados no campo *Message*. Um exemplo é o KB2912390, o qual representa uma atualização de segurança do Windows 7 que visa corrigir uma vulnerabilidade na *API Direct2D*. Portanto, um registro de falha

contendo este valor de KB indica que a *API Direct2D* foi o *componente* de *software* que estava sendo atualizado quando a falha do SWU ocorreu.

No terceiro cenário, quando o *componente* de atualização é analisado, verifica-se o caso em que o mesmo evento de falha ocorreu múltiplas vezes. Portanto, ao invés de comparar apenas o *componente* de atualização, todas as informações presentes nos campos *ProductName* e *Message* das falhas imediatamente antes e/ou imediatamente depois foram comparadas com as da conjunção de referência, a fim de atestar a ocorrência da repetição da mesma falha.

Os exemplos da Tabela 4.14 mostram apenas a investigação das falhas imediatamente antes e depois das conjunções de referência, mas os casos em que as falhas ocorreram somente antes ou somente depois também foram considerados.

Dessa forma, existem três casos para cada cenário. Por exemplo, no primeiro cenário, procura-se por falhas com subtipos diferentes, mas que possuem os mesmos *Error Codes* que os das conjunções de referência quando ocorrem i) imediatamente **antes**, ii) imediatamente **depois** e iii**)** imediatamente **antes** e **depois** da conjunção de referência.

5. RESULTADOS

5.1 Introdução

Este capítulo apresenta os resultados do estudo exploratório realizado nessa pesquisa. Dessa forma, o protocolo criado (ver Capítulo 4) é aplicado à amostra de dados de falha descrita na Seção 3.6.

A Seção 5.2 descreve os resultados da aplicação do protocolo, discute a qualidade dos resultados obtidos, analisa a relação causal entre os **PFC**s mais frequentemente encontrados e, por fim, apresenta o comportamento observado nos *rankings* de **PFC**s durante a execução protocolo.

A Seção 5.3 apresenta os resultados obtidos por meio da execução da adaptação do protocolo de detecção e caracterização das causas de falha de SO. Primeiramente realizou-se uma análise mais detalhada de causas de falhas da categoria SO_{SVC}, especificamente, falhas de atualização de software, dada sua prevalência na amostra de trabalho. Incluiu-se uma análise temporal dessas falhas relacionadas com o serviço Windows Update com o propósito de confirmar os padrões detectados. Por fim, é apresentada uma análise causal das falhas mais frequentemente observadas na amostra de trabalho das categorias OS_{APP} e OS_{KNL}.

5.2 Padrões de Falhas Candidatos e Padrões de Falhas

Primeiramente, esta seção apresenta os resultados obtidos por meio da execução do protocolo de detecção e caraterização dos padrões de falhas de SO descrito na Seção 4.3. Esse protocolo permitiu identificar os eventos de falha de SO por meio dos seus tipos/subtipos e, assim, foram analisados os eventos, de maior recorrência na amostra de trabalho, que ocorreram antes e/ou depois de determinadas falhas de SO. A Tabela 5.1 lista os acrônimos dos tipos/subtipos de falha de SO utilizados nos *rankings* de padrões de falhas candidatos (**PFC**s).

Tabela 5.1. Tipos/Subtipos de falha de SO usados nos *rankings*

Categoria	Tipos [Subtipos] de falhas de SO	Acrônimo
SO_{APP}	explorer.exe	EXP
SO_{SVC}	SWU [Atualização do Windows]	AW7
SO_{SVC}	SWU [Atualização do .Net Framework]	ANF
SO_{SVC}	SWU [Atualização do Internet Explorer]	AIE
SO_{SVC}	SWU [Atualização do MS Office]	AMO

As Tabelas 5.2, 5.3 e 5.4 apresentam os cinco primeiros **PFC**s de cada *ranking* criado. Cada tabela possui três *rankings* de acordo com os **IP**s analisados (2, 4 e 8 horas). A coluna "Rank" contém a posição dos **PFC**s nos *rankings*.

Uma mesma combinação de eventos de falha de SO pode compor **PFC**s em cada um dos 3 *rankings* de uma mesma tabela. Porém, sua posição pode não ser a mesma em cada um dos *rankings*. Por exemplo, a combinação de eventos representada por **PFC**=AFN→AFN ocupa a terceira posição nos *rankings* de 8 e 4 horas da Tabela 5.2, porém, esta mesma combinação ocupa a quarta posição no *ranking* de 2 horas. Portanto, analisando os *rankings* de uma mesma configuração de **PFC** de cima para baixo, ou seja, do *ranking* de 8 para o de 4 horas ou do *ranking* de 4 para o de 2 horas, adicionou-se o símbolo (^) na coluna "Rank" quando o **PFC** ascende de posição e o símbolo (ˇ) quando o **PFC** descende de posição no *ranking* de menor **IP**.

O símbolo (-) foi adicionado nas tabelas para identificar a diminuição da quantidade de ocorrências de um determinado **PFC**, assim como o número de computadores que contenham as ocorrências desse **PFC**. Essa diminuição ocorre quando se compara as ocorrências de um mesmo **PFC** em um *ranking* com **IP** menor com relação a um *ranking* com a mesma configuração, porém, com um **IP** maior. Por exemplo, a combinação de eventos representada por **PFC**=EXP→EXP ocupa a primeira posição nos *rankings* de 8 e 4 horas da Tabela 5.2, mas o número de ocorrências desse **PFC** passa de 98 no *ranking* de 8 horas para 81 no *ranking* de 4 horas.

As colunas "**EFA"**, "**FR"** e "**EFP"** contêm os tipos/subtipos de falha de SO que compõem o **PFC** e a coluna "DP" refere-se ao desvio padrão dos **ΔtA**s e **ΔtP**s**.** O desvio padrão indica o quanto de variação ocorreu com relação à média da proximidade temporal das **FR** em relação aos seus **EFA**s e/ou **EFP**s**.** Portanto, é desejável que esse valor seja o menor possível, o que indica uma maior consistência no tempo de ocorrência das falhas que compõem os **PFC**s.

Tabela 5.2. Cinco primeiras entradas dos *rankings* da configuração PFC=EFA→FR

IP	Rank	EFA	FR	Nº de Grupos com o PFC	% de Computadores com o PFC	Nº de Ocorrências do PFC	DP
8 horas	**1**	EXP	EXP	4	6,3604	98	0,0913
	2	AW7	AW7	4	5,8304	554	0,0645
	3	ANF	ANF	3	2,4735	129	0,1139
	4	AIE	AW7	3	2,4735	28	0,0103
	5	AMO	AMO	3	2,1201	65	0,0542
4 horas	**1**	EXP	EXP	4	6,3604	(-) 81	0,0451
	2	AW7	AW7	4	5,8304	(-) 535	0,0504
	3	ANF	ANF	3	2,4735	(-) 111	0,0596
	4	AIE	AW7	3	2,4735	28	0,0103
	(^) 5	ANF	AW7	3	2,1201	55	0,0091
2 horas	**1**	EXP	EXP	4	(-) 6,1837	(-) 70	0,0257
	2	AW7	AW7	4	(-) 5,6537	(-) 436	0,0176
	(^) 3	AIE	AW7	3	2,4735	28	0,0103
	(ˇ) 4	ANF	ANF	3	(-) 2,2968	(-) 85	0,0251
	5	ANF	AW7	3	2,1201	55	0,0091

Tabela 5.3. Cinco primeiras entradas dos *rankings* da configuração PFC=FR→EFP

IP	Rank	FR	EFP	Nº de Grupos com o PFC	% de Computadores com o PFC	Nº de Ocorrências do PFC	DP
8 horas	**1**	EXP	EXP	4	6,3604	95	0,0957
	2	AW7	AW7	4	6,0071	484	0,1057
	3	ANF	ANF	3	3,5336	151	0,1081
	4	AW7	AIE	3	3,0035	61	0,0852
	5	AMO	AMO	3	2,1201	82	0,0499
4 horas	**1**	EXP	EXP	4	6,3604	(-) 82	0,0460
	2	AW7	AW7	4	6,0071	(-) 398	0,0575
	3	ANF	ANF	3	(-) 3,3569	(-) 121	0,0602
	4	AW7	AIE	3	(-) 2,8269	(-) 56	0,0468
	5	AMO	AMO	3	(-) 1,9435	(-) 79	0,0062
2 horas	**1**	EXP	EXP	4	(-) 6,1837	(-) 70	0,0244
	2	AW7	AW7	4	(-) 5,8304	(-) 291	0,0205
	3	ANF	ANF	3	(-) 3,1802	(-) 95	0,0247
	4	AW7	AIE	3	2,8269	(-) 55	0,0093
	5	AMO	AMO	3	1,9435	79	0,0062

Tabela 5.4. Cinco primeiras entradas dos *rankings* da configuração PFC=EFA→FR→EFP

IP	Rank	EFA	FR	EFP	Nº de Grupos com o PFC	% de Computadores com o PFC	Nº de Ocorrências do PFC	DP
8 horas	**1**	AW7	AW7	AW7	4	5,3004	23337	0,0354
	2	AW7	AW7	AIE	3	2,8269	6099	0,0502
	3	AW7	AW7	ANF	3	2,6502	12203	0,0371
	4	AIE	AW7	AIE	3	2,4735	860	0,0813
	5	AIE	AW7	AW7	3	2,2968	8622	0,0190
4 horas	**1**	AW7	AW7	AW7	4	5,3004	(-) 22702	0,0085
	2	AW7	AW7	AIE	3	2,8269	(-) 5247	0,0156
	3	AW7	AW7	ANF	3	2,6502	(-) 11873	0,0071
	4	AIE	AW7	AIE	3	2,4735	(-) 690	0,0247
	5	AIE	AW7	AW7	3	2,2968	(-) 8579	0,0061
2 horas	**1**	AW7	AW7	AW7	4	5,3004	(-) 22574	0,0021
	2	AW7	AW7	AIE	3	2,8269	(-) 5157	0,0032
	3	AW7	AW7	ANF	3	2,6502	(-) 11834	0,0006
	4	AIE	AW7	AIE	3	(-) 2,2968	(-) 640	0,0069
	5	AIE	AW7	AW7	3	(-) 2,1201	(-) 8537	0,0020

Pode-se observar que os **PFC**s que estão no topo de todos os *rankings* das configurações **PFC**=**EFA**→**FR** (Tabela 5.2) e **PFC**=**FR**→**EFP** (Tabela 5.3) são compostos pelas falhas de explorer.exe (**PFC**=EXP→EXP e **PFC**=EXP→EXP). Como mencionado na Seção 3.6.1, o explorer.exe possui duas principais funcionalidades: gerenciar janelas e manutenção/acesso aos arquivos. Em [Beauvisage 2009], um estudo sobre o uso de computadores domésticos de 1.434 usuários foi realizado durante 19 meses. Os autores verificaram que o explorer.exe foi o único programa utilizado por todos os usuários, obtendo a segunda posição em um *ranking* de tempo de execução das aplicações nos computadores. Portanto, essa intensa utilização do programa pode explicar sua posição de liderança nos *rankings*.

As falhas de Atualização do Windows (AW7) compuseram os **PFC**s presentes na segunda posição de todos os *rankings* das Tabelas 5.2 e 5.3. Da mesma forma que os **PFC**s compostos pelas falhas do explorer.exe, tanto as **FR**s quanto os **EFA**s para a configuração **PFC**=**EFA**→**FR** e os **EFP**s para a configuração **PFC**=**FR**→**EFP** possuem os mesmos tipos/subtipos. Nesse caso, todos os **PFC**s foram compostos pelo mesmo subtipo de falha do SWU (AW7). Portanto, os **PFC**s observados na segunda posição desses *rankings* foram **PFC**=AW7→AW7 e **PFC**=AW7→AW7.

Na Tabela 5.2, a terceira posição dos *rankings* de 8 horas e 4 horas é ocupada pelos eventos de falha de Atualização do .Net Framework, os quais ocorreram tanto como **EFA** quanto **FR** (**PFC**=AFN→AFN). No *ranking* de 2 horas, esse **PFC** ocupa a quarta posição, devido à diminuição de sua ocorrência em alguns computadores nesse intervalo de pesquisa. A terceira posição no *ranking* de 2 horas é ocupada pela combinação de eventos representada por **PFC**=AIE→AW7. Essa combinação de eventos ocupou a quarta posição nos *rankings* de 8 e 4 horas.

A quinta posição do *ranking* de 8 horas (ver Tabela 5.2) é ocupada pela combinação de falhas de Atualização do MS Office (**PFC**=AMO→AMO). Para os *rankings* de 4 e 2 horas, a quinta posição foi ocupada pela combinação representada por **PFC**=AIE→AW7.

Na Tabela 5.2, a terceiro posição em todos os *rankings* é ocupada pelo **PFC**=ANF→ANF, ou seja, uma combinação de falhas de Atualização do .Net Framework. Da mesma forma, a quarta posição também é ocupada pelo mesmo **PFC** em todos os *rankings* (**PFC**=AW7→AIE), ou seja, Atualização do Windows como **FR** e Atualização do Internet Explorer como **EFP**. E a quinta posição é ocupada pelas falhas de Atualização do MS Office (**PFC**=AMO→AMO), também presente em todos os *rankings* na mesma posição.

A Tabela 5.3 contém os *rankings* para os **PFC**s com a configuração **PFC**=**EFP**→**FR**→**EFP**. A combinação de eventos que compõem os **PFC**s da primeira posição dos *rankings*, **PFC**=AW7→AW7→AW7, segue o mesmo padrão dos primeiros colocados dos *rankings* das outras duas configurações (**PFC**=**EFA**→**FR** e **PFC**=**FR**→**EFP**). Isto é, o tipo/subtipo da **FR** é o mesmo tipo/subtipo de seus **EFA**s e **EFP**s.

Isso não ocorre para as outras posições dessa tabela. Observou-se que todos os outros **PFC**s, os quais ocupam as outras posições dos *rankings*, possuem a mesma falha de referência (Atualização do Windows) e todos os seus **EFA**s e **EFP**s também pertencem à categoria Serviço de SO e são falhas de atualização de *software*. Portanto, todos os **PFC**s nessa tabela são compostos pelo subtipo AW7 e por outros subtipos do tipo SWU. Esses resultados sugerem que as falhas de AW7 podem ter alguma relação com outras falhas de atualização de *software*.

Todos os **PFC**s da Tabela 5.3 se mantiveram nas mesmas posições dos *rankings*, indicando fortes evidências que permite considerá-los padrões de falhas genuínos (**PF**s).

Comparando todos os *rankings*, verificou-se, por diversas vezes, a ocorrência de **EFA**s e/ou **EFP**s com os tipos/subtipos idênticos aos tipos/subtipos de suas **FR**s, principalmente para as configurações **PFC**=**EFA**→**FR** e **PFC**=**FR**→**EFP**. Este padrão sugere duas possíveis situações:

1. O mesmo componente do SO (aplicação/serviço/subsistema do *Kernel*) falhou múltiplas vezes, uma de cada vez, em execuções repetidas. Por exemplo, a aplicação de SO explorer.exe, a qual executa como uma única instância (processo), falha e precisa ser reinicializada e, subsequentemente, a nova instância falha novamente.

2. A execução de várias instâncias simultâneas do mesmo componente do SO (aplicação/serviço/subsistema do *Kernel*), em que duas ou mais instâncias falham ao mesmo tempo ou em intervalos de tempo muito próximos. Por exemplo, o serviço svchost.exe executa como várias instâncias e algumas dessas instâncias poderiam falhar dentro de um curto intervalo de tempo ou, simplesmente, ao mesmo tempo.

Os mesmos eventos de falha que compõem os **PFC**s de todos os *rankings* da Tabela 5.2, também compõem os **PFC**s de todos os *rankings* da Tabela 5.3. Isto ocorreu devido ao grande número de ocorrências do mesmo tipo/subtipo de falha de SO em um curto intervalo de tempo. Por exemplo, ao analisar duas ocorrências de falhas consecutivas do tipo explorer.exe, com um minuto de intervalo, explorer.exe$^{12:54}$ e explorer.exe$^{12:55}$. Dado que a **FR**=explorer.exe e o **IP**=2 horas, o protocolo irá detectar um **PFC** com a representação **PFC**=explorer.exe$^{12:54}$→explorer.exe$^{12:55}$ quando **PFC**s com a configuração **PFC**=**EFA**→**FR** são analisados (Tabela 5.2) e irá detectar um outro **PFC** com a representação **PFC**=explorer.exe$^{12:54}$→explorer.exe$^{12:55}$ quando **PFC**s com a configuração **PFC**=**FR**→**EFP** são analisados (Tabela 5.3). Portanto, as mesmas ocorrências de falhas são analisadas para as duas configurações, fazendo com que a composição dos **PFC**s seja similar nas duas tabelas.

Esse comportamento explica o motivo pelo qual o numero de ocorrências de **PFC**s (542.016) é maior do que o número total de falhas (7.007). Uma vez que uma determinada ocorrência de falha pode ser definida como **FR** uma única vez para cada um dos *rankings*, entretanto, ela pode ser caracterizada como **EFA** ou **EFP** inúmeras vezes.

5.2.1 Qualidade dos Resultados

O protocolo criado tem como objetivo detectar e caracterizar padrões de falhas consistentes. Portanto, definiu-se, anteriormente, que os **PFC**s devem ser detectados em pelo menos três dos quatro grupos da amostra de trabalho (ver Seção 4.3.3). Assim, o primeiro critério analisado para incluir um determinado **PFC** nos *rankings* é a quantidade de grupos que contenha pelo menos uma ocorrência do **PFC**.

Apesar de não explícitos nas Tabelas 5.2, 5.3 e 5.4, foram criados três formatos para contabilizar o número de grupos com ocorrências dos **PFC**s, sendo eles: 4-4, 4-3 e 3-3. Em todos os três formatos, o primeiro número representa a quantidade de grupos com pelo menos uma ocorrência da **FR** que compõe o **PFC, e** o segundo número indica a quantidade de grupos com ao menos uma ocorrência do **PFC**. Por exemplo, o **PFC**=AIE→AW7 para o **IP**=8 horas possui o formato 4-3, ou seja, a **FR**=AW7 foi observada em quatro grupos (ver Tabela 4.2 da Seção 4.3.1) e o **PFC**=AIE→AW7 em três grupos.

Apenas os formatos 4-4 e 4-3 ocorreram nas Tabelas 5.2, 5.3 e 5.4, sendo que somente os dois primeiros **PFCs** de todos os *rankings* das Tabelas 5.2 e 5.3, e o primeiro de todos os *rankings* da Tabela 5.4, possuem o formato 4-4.

Com base nos valores obtidos da coluna "DP" das Tabelas 5.2, 5.3 e 5.4, foi observado que os **PFC**s que ocuparam a primeira posição nos *rankings* não possuem os menores desvios padrões em relação aos seus **ΔtA**s e **ΔtP**s. Apesar disso, a análise dos desvios padrões de todos os **PFC**s mostra uma diferença pequena entre os maiores e os menores desvios padrões. Esta baixa variabilidade nos tempos de ocorrência da **FR** e seus **EFA**s e/ou **EFP**s, para os **PFC**s detectados, indica a boa consistência de suas ocorrências por meio dos diferentes grupos de computadores analisados.

Embora o conjunto de dados da amostra de trabalho apresente falhas da categoria *Kernel*, principalmente no grupo G1, com mais de 80% de ocorrências dessas falhas, foram identificados apenas dois **PFC**s pertencentes a essa categoria.

Por meio da análise de todas as posições dos *rankings* com as configurações **PFC=EFA→FR** e **PFC=FR→EFP**, no **IP**=8 horas, os seguintes **PFC**s da categoria *Kernel* foram detectados.

- **PFC**=DRIVER_POWER_STATE_FAILURE→DRIVER_POWER_STATE_FAILURE
- **PFC**=DRIVER_POWER_STATE_FAILURE→DRIVER_POWER_STATE_FAILURE

As falhas do tipo DRIVER_POWER_STATE_FAILURE sempre ocorrem quando um *device driver* ou uma rotina do sistema operacional executando em nível de *Kernel* entra em um estado inválido[Russinovich *et al.* 2009].

As Tabelas 5.2, 5.3 e 5.4 listam 45 **PFC**s com 153.511 ocorrências. Devido à diversidade de grupos e da quantidade de **PFC**s encontrados, esses **PFC**s foram classificados como padrões de falhas (**PF**s). A Tabela 5.5 mostra a quantidade de ocorrências desses **PF**s em relação aos intervalos de pesquisa e configurações de padrões analisados. Nota-se, por meio da coluna "Total" que a maior quantidade de ocorrências de **PF**s é da configuração **PF=EFA→FR→EFP**. Isso indica que as falhas analisadas possuem uma ordem temporal específica de ocorrer antes e depois de determinadas falhas.

Tabela 5.5. Número de ocorrência dos PFs

IP	PF=EFA→FR	PF=FE→EFP	PF=EFA→FR→EFP	Total
8 horas	874	873	51.121	52.868
4 horas	810	736	49.091	50.637
2 horas	674	590	48.742	50.006
Total	2.358	2.199	148.954	153.511

5.2.2 Relação Entre as Causas das Falhas e os Padrões de Falhas Candidatos

Após uma análise de não apenas os cinco primeiros colocados dos *rankings*, mas dos *rankings* completos (296 **PFC**s com 542.016 ocorrências), observou-se que a maioria das ocorrências dos **PFC**s (72 **PFC**s com 308.955 ocorrências, 57% das ocorrências), é composta pela mesma **FR**, o subtipo do SWU Atualização do Windows (AW7), sendo que seus **EFA**s **e**/ou **EFP**s também são subtipos do SWU.

Por esse motivo, concluiu-se que além do AW7 ser o segundo subtipo de maior ocorrência na amostra de falha (ver Tabela 3.13 da Seção 3.6.1), ele também se relaciona com a maioria dos subtipos de falha do SWU.

Dada a sua importância para o estudo, o subtipo AW7 foi investigado mais detalhadamente. Para isso, analisou-se a causa das ocorrências desse subtipo de falha com base nos *Error Codes* presentes no campo *Message* desses registros de falha.

Como já explicado na Seção 4.5.1, as três causas (*Error Codes*) mais frequentes de falhas de AW7 na amostra de trabalho são:

1. 0x80070643: indica que um mau funcionamento do .Net Framework causou a falha durante o processo de instalação da atualização de *software*.

2. 0x800706ba: indica que a indisponibilidade do servidor RPC (*Remote Procedure Call*) causou uma falha no SWU.

3. 0x800b0100: indica que os arquivos necessários para o SWU atualizar um determinado *software* estão faltando ou estão corrompidos.

Essa análise das causas mais frequentes das falhas de AW7 foi realizada sobre as 1.164 ocorrências individuais das falhas desse subtipo, ou seja, independente de suas contribuições para compor os padrões de falhas. Posteriormente, realizou-se também uma análise das ocorrências das falhas com o subtipo AW7, considerando sua contribuição (influência) para compor os **PFC**s.

Inicialmente, todas as ocorrências dos **PFC**s encontradas na amostra de trabalho, isto é, 542.016 ocorrências**,** foram filtradas. Apenas as ocorrências dos **PFC**s compostos pelas **FR**s iguais a um dos três *Error Codes* citados anteriormente e com o subtipo AW7, ou seja, as conjunções *Error Code* e subtipo do SWU: 0x80070643 & AW7, 0x800706ba & AW7 e 0x800b0100 & AW7.

A primeira e a terceira conjunções aparecem em apenas 1725 (0,73%) das ocorrências dos **PFC**s. Isto é uma evidência de que as falhas causadas por esses dois *Error Codes* ocorrem isoladas de outras falhas de SO com relação ao tempo.

Por outro lado, analisando a segunda conjunção (0x800706ba & AW7) como **FR**, verificou-se que o *Error Code* 0x800706ba representou a causa de falha mais influente do subtipo AW7, estando presente em 235.973 (76.38%) dos **PFC**s com esta composição. Este é um caso de dependência entre as falhas, pois esse *Error Code* indica que o serviço Windows *Update* (SWU) falhou devido à indisponibilidade ou falha do serviço RPC, podendo causar a falha de todas as atualizações de *software* sendo executadas.

Em seguida, buscou-se pela segunda e terceira causas de falha mais influentes com o subtipo AW7 como **FR**. O segundo *Error Code* mais influente do subtipo AW7 é o 0xc8000710, ele esteve presente em 56.517 (18.30%) ocorrências de **PFC**s. Esse *Error Code*, descrito na Seção 4.5.1, indica que a falta de espaço em disco durante o *download* de novas atualizações de *software* causou a falha do SWU.

O *Error Code* 0x800f0902, presente em 7.962 (2,58%) dos **PFC**s, foi a terceira causa de falha mais influente para o subtipo AW7. Uma falha com esse *Error Code* ocorre quando o Windows está realizando outra operação de serviço, impedindo que a rotina de atualização seja concluída [Microsoft 2016r].

A Tabela 5.6 apresenta a quantidade de **PFC**s com as conjunções de AW7 de maior influência (0x800706ba – AW7, 0xc8000710 – AW7 e 0x800f0902 – AW7) como **FR** para cada uma das três possíveis configurações de **PFC**s (**PFC=EFA→FR**, **PFC=FR→EFP** e **PFC=EFA→FR→EFP**).

Observou-se que a quantidade de **PFC**s para a configuração **PFC=EFA→FR→EFP** é maior do que para as outras configurações. Analisando os **EFA**s e/ou **EFP**s dos **PFC**s compostos pelas conjunções de maior influência com o subtipo AW7 como **FR**, verificou-se que a maioria dos **EFA**s e/ou **EFP**s possuem as mesmas causas que a **FR**, ou seja, os mesmos *Error Codes*. Isso indica uma causa em comum de várias falhas de atualização de *software* (SWU), dentro de um determinado intervalo de tempo.

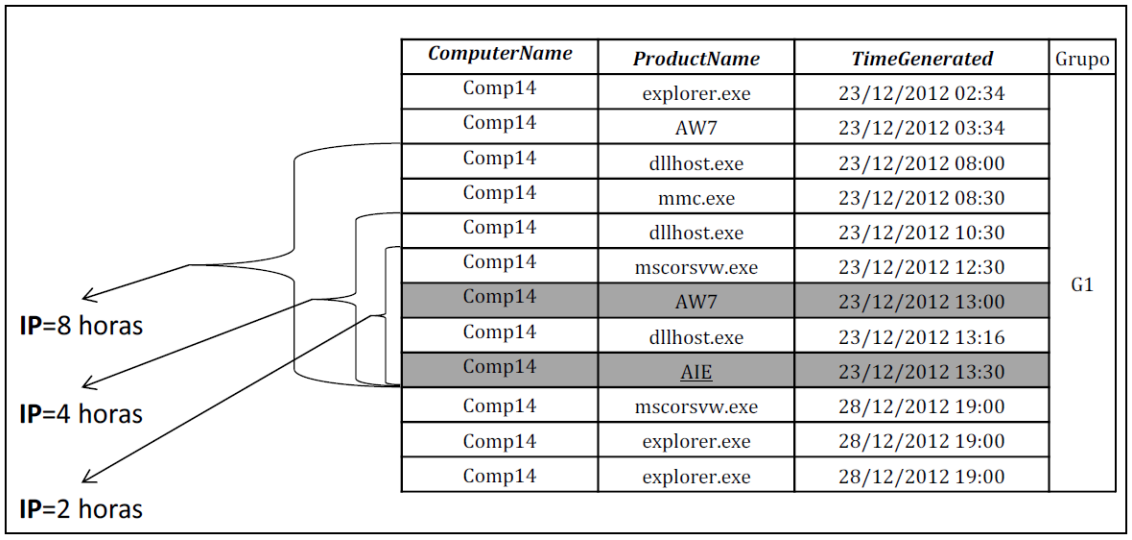
Tabela 5.6. Quantidade de PFCs com as conjunções de AW7 de maior influência como FR

PFC=EFA→FR					
IP	**Total**	**AW7**	**0x800706ba & AW7**	**0xc8000710 & AW7**	**0x800f0902 & AW7**
8 horas	2.332	690	307	0	69
4 horas	2.082	671	303	0	69
2 horas	1.573	572	303	0	69
PFC=FR→EFP					
IP	**Total**	**AW7**	**0x800706ba & AW7**	**0xc8000710 & AW7**	**0x800f0902 & AW7**
8 horas	2.328	613	11	0	53
4 horas	2.010	470	11	0	53
2 horas	1.496	362	11	0	53
PFC=EFA→FR→EFP					
IP	**Total**	**AW7**	**0x800706ba & AW7**	**0xc8000710 & AW7**	**0x800f0902 & AW7**
8 horas	186.905	103.723	79.751	18.839	2.532
4 horas	172.177	101.152	77.638	18.839	2.532
2 horas	171.113	100.702	77.638	18.839	2.532
Total					
IP	**Total**	**AW7**	**0x800706ba & AW7**	**0xc8000710 & AW7**	**0x800f0902 & AW7**
Todos	542.016	308.955	235.973	56.517	7.962

5.2.3 Comportamento dos PFCs ao Decrementar os IPs

Uma mesma combinação de eventos de falha de SO pode compor **PFC**s em cada um dos 3 *rankings* de uma mesma configuração (ex. *rankings* **PFC=EFA→FR** (8h)**, PFC=EFA→FR** (4h) **e PFC=EFA→FR** (2h)). Esta seção apresenta uma explicação sobre o comportamento dessas combinações de eventos quando comparadas com elas mesmas nos diferentes *rankings* que estão presentes. Espera-se que o número de computadores com os mesmos **PFC**s, assim como o número de ocorrências dos **PFC**s, permaneça constante ou decremente em curtos **IP**s em comparação com os maiores (ex. IP de 2 horas com relação ao de 4 horas).

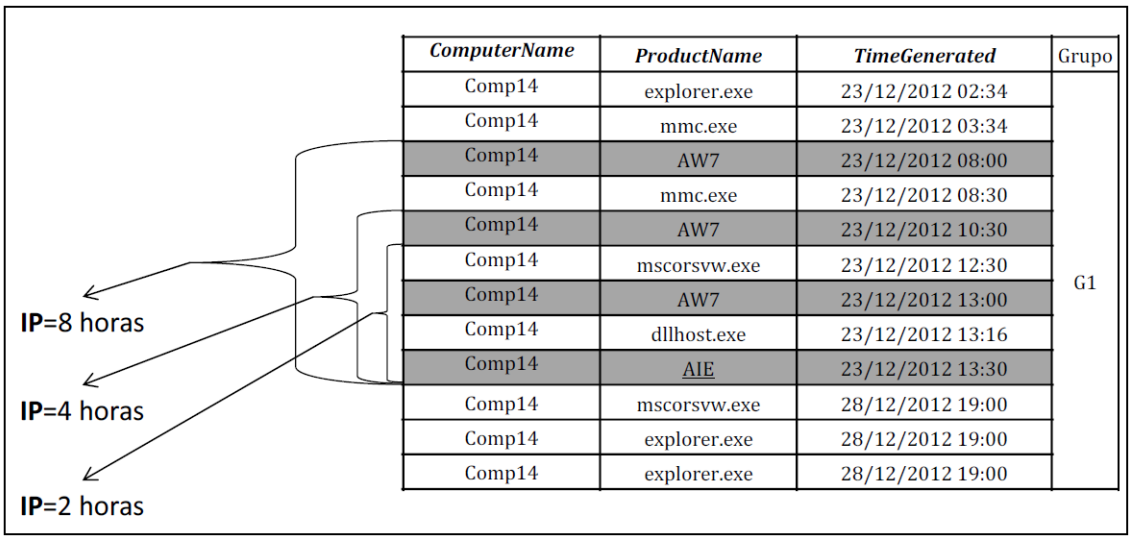
O caso em que a quantidade de ocorrências do mesmo **PFC** permanece constante quando o **IP** é decrementado ocorre devido à proximidade do tempo de ocorrência de uma determinada **FR** em relação aos seus **EFA**s e **EFP**s. Portanto, os **EFA**s e **EFP**s ocorreram dentro de um intervalo de pesquisa menor ou igual ao menor **IP** analisado (**IP**=2 horas) com relação à **FR**. Dessa forma, mesmo quando os intervalos de pesquisa foram reduzidos (8 para 4 horas e 4 para 2 horas), a quantidade de ocorrências dos **PFC**s, nesses casos, permaneceu a mesma. A Figura 5.1 apresenta um exemplo para esse cenário com a combinação de eventos representada por **PFC**=AW7→AIE. Nesse exemplo, existe apenas uma ocorrência do **PFC**. A diferença temporal entre o **EFA**=AW7$^{13:00}$ e a **FR**=AIE$^{13:30}$ é de 0,5 horas. Portanto, esse **PFC** está presente em todos **IP**s (8, 4 e 2 horas).



ComputerName	*ProductName*	*TimeGenerated*	Grupo
Comp14	explorer.exe	23/12/2012 02:34	G1
Comp14	AW7	23/12/2012 03:34	
Comp14	dllhost.exe	23/12/2012 08:00	
Comp14	mmc.exe	23/12/2012 08:30	
Comp14	dllhost.exe	23/12/2012 10:30	
Comp14	mscorsvw.exe	23/12/2012 12:30	
Comp14	AW7	23/12/2012 13:00	
Comp14	dllhost.exe	23/12/2012 13:16	
Comp14	AIE	23/12/2012 13:30	
Comp14	mscorsvw.exe	28/12/2012 19:00	
Comp14	explorer.exe	28/12/2012 19:00	
Comp14	explorer.exe	28/12/2012 19:00	

Figura 5.1. Quantidade de ocorrências de PFCs permanece constante quando o IP é decrementado.

Existe também o caso de diminuição da quantidade de ocorrências de um determinado **PFC**, assim como, o número de computadores com as ocorrências desse **PFC**, quando o **IP** é decrementado. Isso indica que a diferença de tempo entre a ocorrência da **FR** e de seus **EFAs** e **EFPs** é maior do que o **IP** sendo analisado. A Figura 5.2 apresenta um exemplo para esse caso, com a combinação de eventos representada por **PFC**=AW7→AIE. Quando se calcula a quantidade de ocorrências desse **PFC**, a quantidade muda de acordo com o **IP** analisado. Por exemplo, para o **IP**=8 horas, existem 3 ocorrências do **PFC** investigado, com combinação de eventos representada pelos **EFAs** AW7$^{08:00}$, AW7$^{10:30}$ e AW7$^{13:00}$, todos com relação à **FR**=AIE$^{13:30}$. Ao analisar a quantidade de ocorrências desse **PFC** para o **IP**=4 horas, a quantidade diminui para 2 ocorrências, com os **EFAs** AW7$^{10:30}$ e AW7$^{13:00}$ em relação à **FR**=AIE$^{13:30}$. E para o **IP**=2 horas, existe apenas uma ocorrência, **PFC**=AW7$^{13:00}$→AIE$^{13:30}$.



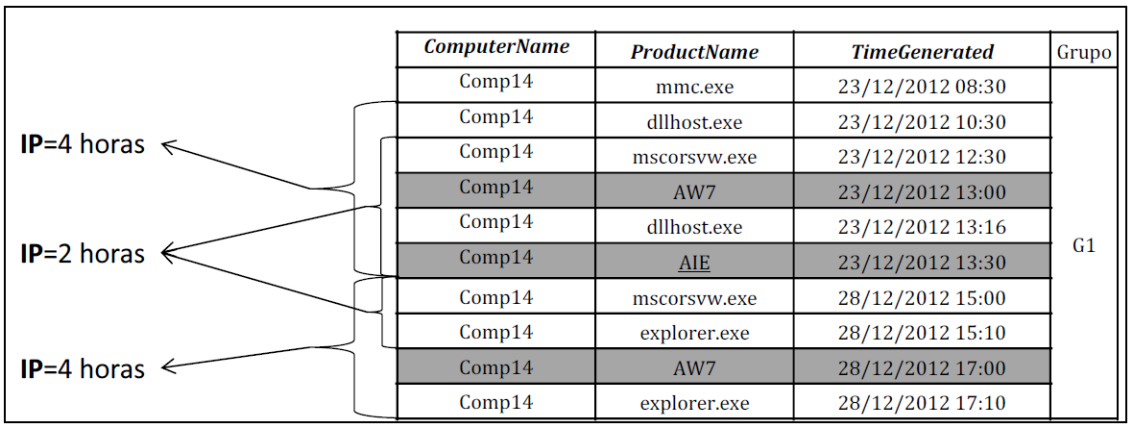
ComputerName	*ProductName*	*TimeGenerated*	Grupo
Comp14	explorer.exe	23/12/2012 02:34	G1
Comp14	mmc.exe	23/12/2012 03:34	
Comp14	AW7	23/12/2012 08:00	
Comp14	mmc.exe	23/12/2012 08:30	
Comp14	AW7	23/12/2012 10:30	
Comp14	mscorsvw.exe	23/12/2012 12:30	
Comp14	AW7	23/12/2012 13:00	
Comp14	dllhost.exe	23/12/2012 13:16	
Comp14	AIE	23/12/2012 13:30	
Comp14	mscorsvw.exe	28/12/2012 19:00	
Comp14	explorer.exe	28/12/2012 19:00	
Comp14	explorer.exe	28/12/2012 19:00	

Figura 5.2. Quantidade de ocorrências de PFCs diminui quando o IP é decrementado.

Além desses dois comportamentos descritos anteriormente, observou-se também o aumento de ocorrências de **PFCs** e de computadores com essas ocorrências quando os **IPs** foram decrementados durante a aplicação do protocolo. Esse aumento pode ocorrer apenas para as configurações **PFC**=**EFA**→**FR** e **PFC**=**FR**→**EFP**.

O comportamento de aumento de ocorrências de **PFCs** foi causado pela ocorrência de padrões com a configuração **PFC**=**EFA**→**FR**→**EFP** nos **IPs** mais longos definidos nesse estudo, o que previne a ocorrência das outras duas configurações (**PFC**=**EFA**→**FR** e **PFC**=**FR**→**EFP**). Esse aumento ocorre quando o **IP** é reduzido (ex. 8 para 4 horas), e uma combinação de eventos com a configuração **PFC**=**EFA**→**FR**→**EFP**, a qual ocorreu para o maior **IP** (8 horas), não ocorre para o menor **IP** (4 horas). Portanto, a combinação de eventos com a configuração **PFC**=**EFA**→**FR**→**EFP** é desfeita para o menor **IP**, o que possibilita identificar combinações de eventos com a configuração **PFC**=**EFA**→**FR** ou **PFC**=**FR**→**EFP**. Por consequência, há o aumento de combinações de eventos para as configurações **PFC**=**EFA**→**FR** ou **PFC**=**FR**→**EFP,** causando também o aumento no número de computadores com esses **PFCs.** Esse comportamento foi observado nos resultados deste estudo apenas para a configuração **PFC**=**FR**→**EFP** para o **IP** de 4 horas nas combinações de eventos representadas por **PFC**=AIE→AIE e **PFC**=AIE→AW7. Um exemplo é apresentado na Figura 5.3, para o caso em que a combinação de eventos representada por **PFC**=AW7→AIE é analisada.

Nota-se nesse exemplo, que para o **IP**=4 horas existe uma ocorrência do **PFC**=$AW7^{13:00}$→$AIE^{13:30}$→$AW7^{17:00}$ e, de acordo com o protocolo (ver Seção 4.3.4), não existe nenhuma ocorrência do padrão sendo investigado (**PFC**=AW7→AIE). Porém, para o **IP**=2 horas o **PFC** com configuração **PFC**=**EFA**→**FR**→**EFP** se desfaz, pois o **EFP**= $AW7^{17:00}$ não se encontra dentro desse novo **IP** analisado, formando uma ocorrência do **PFC**=$AW7^{13:00}$→$AIE^{13:30}$.



ComputerName	*ProductName*	*TimeGenerated*	Grupo
Comp14	mmc.exe	23/12/2012 08:30	G1
Comp14	dllhost.exe	23/12/2012 10:30	
Comp14	mscorsvw.exe	23/12/2012 12:30	
Comp14	AW7	23/12/2012 13:00	
Comp14	dllhost.exe	23/12/2012 13:16	
Comp14	AIE	23/12/2012 13:30	
Comp14	mscorsvw.exe	28/12/2012 15:00	
Comp14	explorer.exe	28/12/2012 15:10	
Comp14	AW7	28/12/2012 17:00	
Comp14	explorer.exe	28/12/2012 17:10	

Figura 5.3. Quantidade de ocorrências de PFCs aumenta quando o IP é decrementado.

5.3 Padrões entre as causas das falhas de Sistema Operacional

5.3.1 Análise de Causalidade Entre as Falhas do SWU

Esta seção apresenta os resultados do protocolo usado para a detecção e caracterização das causas de falha de SO. Foram investigadas as relações causais entre as falhas e seus padrões de causalidade.

Primeiramente, foram analisadas as falhas do SWU causadas pelos *Error Codes* mais frequentes na amostra de trabalho. Portanto, foram investigados os eventos de falha que ocorreram imediatamente antes e/ou imediatamente depois dos eventos de falha com essas conjunções (*Error Codes* & subtipos do SWU), com base nos três cenários desenvolvidos para analisar padrões das causas de falha do SWU (ver Seção 4.5.3).

A Tabela 5.6 mostra os resultados obtidos da investigação dos padrões de causalidade das falhas do SWU para cada cenário, considerando as causas de falha (*Error Codes*) mais frequentes listadas na Tabela 4.11 da Seção 4.5.1. As colunas 2, 3 e 4 listam as porcentagens de falhas encontradas para os cenários 1, 2 e 3, respectivamente, em relação ao número total de falhas na categoria SO_{SVC} encontradas para as conjunções analisadas (conjunções de referência).

A coluna SWU(1) representa o primeiro cenário. Quanto maior a porcentagem dessa coluna, mais forte é a evidência de uma causa comum de falhas de diferentes subtipos do SWU.

A coluna SWU(2) representa o segundo cenário. Neste cenário, as falhas podem ter ocorrido quando os componentes de atualização eram *i*) diferentes ou *ii*) iguais ao da conjunção de referência.

O primeiro caso (*i*) do SWU(2) representa as falhas do SWU quando este serviço executava múltiplas atualizações para o mesmo subtipo, entretanto, para um componente de *software* diferente. Isso acontece quando o SWU está processando um conjunto de atualizações do mesmo subtipo e uma falha do SWU impede a instalação de todas essas atualizações. Diferentemente, no segundo caso (*ii*), uma única atualização de *software* falhou múltiplas vezes.

A coluna SWU(3) foi criada para tabular a porcentagem de eventos de falha do segundo caso (*ii*) do segundo cenário (SWU(2)). O valor dessa coluna permite avaliar se o SWU tentou, por diversas vezes, e sem sucesso, instalar a mesma atualização de *software*.

Tabela 5.6. Resultado dos três cenários analisados

Atualização do Internet Explorer (AIE)			
Error Codes	**SWU(1)**	**SWU(2)**	**SWU(3)**
0x80070643	2,00%	97,30%	97,30%
0x80242016	28,92%	38,55%	0,0%
0x80073712	2,82%	95,77%	95,77%
Atualização do Windows (AW7)			
Error Codes	**SWU(1)**	**SWU(2)**	**SWU(3)**
0x80070643	0,26%	94,13%	94,13%
0x800706ba	23,29%	75,54%	0,0%
0x800b0100	0,0%	55,14%	25,23%
Atualização do .Net Framework (ANF)			
Error Codes	**SWU(1)**	**SWU(2)**	**SWU(3)**
0x800b0100	0,0%	99,67%	99,67%
0x800706ba	75,28%	20,22%	0,0%
0xc8000710	45,83%	37,50%	0,0%
Atualização do MS Office (AMO)			
Error Codes	**SWU(1)**	**SWU(2)**	**SWU(3)**
0x80070643	6,45%	82,26%	80,65%
0x80070652	7,62%	81,90%	0,0%
0x800706ba	68,75%	29,69%	0,0%
Geral - 23 subtipos (SWU)			
Error Codes	**SWU(1)**	**SWU(2)**	**SWU(3)**
0x80070643	0,72%	94,49%	91,84%
0x800706ba	39,20%	59,01%	0,0%
0x800b0100	0,0%	81,24%	68,86%

Com base nos valores listados na Tabela 5.6, observam-se diversos casos em que as colunas SWU(2) e SWU(3) possuem as mesmas porcentagens. Isto indica que, em todos esses casos, as falhas imediatamente antes e/ou imediatamente depois, além de terem as mesmas causas e pertencerem aos mesmos subtipos, ocorreram durante a atualização do mesmo componente de *software* que o da conjunção de referência, isto é, a mesma atualização falhou repetidamente.

Particularmente, observaram-se diferentes valores na coluna SWU(3). Quando as conjunções possuem 0% nesta coluna, isso evidencia uma causa comum em diferentes falhas do SWU. Neste caso, a falha na operação do SWU provocou a falha de diferentes atualizações de *software,* ao mesmo tempo, pois o SWU normalmente realiza o *download* de diversas atualizações simultaneamente e as executa sequencialmente. Por exemplo, as falhas do SWU com o *Error Code* 0x800706ba indicam que o serviço RPC se tornou indisponível, impedindo o SWU de realizar quaisquer atualizações, causando as falhas de todas as atualizações em andamento.

Também, observou-se que todas as causas de falha (*Error Codes*) com valores diferentes de 0% na coluna SWU(3) são relacionadas com o processo de instalação de atualizações de *software*. O único *Error Code* relacionado com o processo de instalação que não apresenta valores diferentes de 0% na coluna SWU(3) foi o 0x80070652. Esse *Error Code* indica que, quando o processo de instalação da atualização foi iniciado, existe outra instalação em andamento e ambas não podem ser executadas simultaneamente.

Buscando por mais evidências de que as causas de falha relacionadas com o processo de instalação de atualizações de *software* induziram o SWU a tentar, diversas vezes, e sem sucesso, instalar a mesma atualização, analisou-se a quantidade de vezes em que a mesma conjunção ocorreu no mesmo computador. A Tabela 5.7 apresenta o resultado dessa análise. As colunas 2, 3, e 4 mostram, respectivamente, a porcentagem total dos eventos de falha com as conjunções mais frequentes na amostra de trabalho (% de Eventos), a porcentagem dessas conjunções com apenas uma ocorrência por computador (1:1) e a porcentagem dessas conjunções com mais de uma ocorrência por computador (N:1). A quinta coluna indica a porcentagem que uma determinada falha se repete em um mesmo computador (Repetição:1), ou seja, este percentual representa a quantidade de ocorrências de múltiplas falhas do mesmo tipo, subtipo, componente de atualização e com o mesmo *Error Code*, as quais ocorrem de forma sucessiva.

Tabela 5.7. Porcentagem de falhas do SWU repetidas

Atualização do Internet Explorer (AIE)				
Error Codes	**% de Eventos de Falha (Total = 1.586)**	**1:1**	**N:1**	**Repetição:1**
0x80070643	91,42%	0,63%	90,73%	90,73%
0x80242016	3,78%	0,82%	2,96%	0,0%
0x80073712	2,27%	0,06%	2,21%	2,21%
Atualização do Windows (AW7)				
Error Codes	**% de Eventos de Falha (Total = 1.164)**	**1:1**	**N:1**	**Repetição:1**
0x80070643	33,42%	0,26%	33,16%	33,16%
0x800706ba	25,95%	0,26%	25,43%	0,17%
0x800b0100	9,62%	0,09%	9,54%	8,85%
Atualização do .Net Framework (ANF)				
Error Codes	**% de Eventos de Falha (Total = 309)**	**1:1**	**N:1**	**Repetição:1**
0x800b0100	49,84%	0,32%	49,51%	49,51%
0x800706ba	14,89%	0,97%	13,92%	0,0%
0xc8000710	7,77%	0,0%	7,77%	2,59%
Atualização do MS Office (AMO)				
Error Codes	**% de Eventos de Falha (Total = 266)**	**1:1**	**N:1**	**Repetição:1**
0x80070643	36,47%	14,29%	22,18%	21,80%
0x80070652	24,81%	4,51%	20,30%	0,0%
0x800706ba	24,44%	0,38%	24,06%	0,0%
Geral - 23 subtipos (SWU)				
Error Codes	**% de Eventos de Falha (Total = 3.737)**	**1:1**	**N:1**	**Repetição:1**
0x80070643	61,20%	2,01%	59,19%	56,86%
0x800706ba	11,99%	0,05%	11,93%	0,05%
0x800b0100	7,12%	0,05%	7,06%	6,85%

Com base nos resultados listados na Tabela 5.7, concluiu-se que, em geral, a porcentagem de eventos de falha que possuem mais de uma ocorrência no mesmo computador é maior do que a porcentagem de falhas com apenas uma ocorrência. Isto indica que as causas de falha analisadas são consistentemente observadas.

Comparando-se as colunas 4 e 5, na Tabela 5.7, mais evidências que apoiam os achados anteriores foram encontradas, as quais indicam que as falhas que ocorreram durante o processo de atualização de *software* (ex. falhas causadas pelos *Error Codes* 0x80070643, 0x800b0100, e 0x80073712) tiveram suas ocorrências repetidas. Nesses casos, a porcentagem da quinta coluna é próxima da quarta coluna, indicando que quase todas as ocorrências de falha da mesma conjunção, presente no mesmo computador, são repetições da mesma falha. Esse achado sugere que o SWU foi programado para executar as operações de atualização de *software* automaticamente, resultando na recorrência do mesmo evento de falha múltiplas e sucessivas vezes.

Ressalta-se que apenas as falhas causadas pelo *Error Code* 0x80070652 não seguiram o padrão aqui discutido, observado em falhas de atualização de *software* que ocorreram durante o processo de instalação. As falhas com esse *Error Code* ocorreram quando múltiplas instalações tentaram executar ao mesmo tempo. Portanto, uma possível explicação do motivo pelo qual essas falhas não seguiram o padrão de se repetirem é que, quando a instalação de uma determinada atualização de *software* falha devido a esse *Error Code*, o SWU é capaz de reinstalar essa atualização que falhou anteriormente no momento em que as instalações concorrentes terminarem.

5.3.2 Análise Temporal das Falhas do SWU

Foi realizada uma análise temporal das falhas do SWU com o propósito de identificar as diferenças entre as conjunções (*Error Codes* e subtipo do SWU) de referência e as falhas que ocorreram imediatamente antes e/ou imediatamente depois destas conjunções. Essa análise foi realizada para identificar se as falhas que ocorreram durante o processo de atualização de *software* aconteceram repetidamente durante um curto período tempo.

A Tabela 5.8 apresenta o método utilizado para calcular essa diferença temporal. Para esse propósito, foi necessário utilizar o campo *TimeGenerated* dos registros de falha (ver Seção 3.3). A linha sombreada representa o evento de falha da conjunção de referência sendo analisada. Essa linha não possui valor para a diferença temporal (N/A), pois nesta análise se buscou apenas pela diferença temporal das falhas que ocorreram imediatamente antes e/ou imediatamente depois das conjunções de referência.

Tabela 5.8. Exemplo de diferenças temporais entre os eventos de falha

Error Code	*ProductName*	*TimeGenerated*	Diferença
0x80242016	Atualização do Internet Explorer	12/30/2012 08:00	2 horas
0x80070643	Atualização do Windows	12/30/2012 10:00	N/A
0x80242016	Atualização do Internet Explorer	12/30/2012 11:00	1 hora

Com base nessa abordagem, os histogramas das diferenças de tempo foram calculados. Esses histogramas foram utilizados para analisar os três cenários discutidos na Seção 4.5.3, isto é, SWU(1), SWU(2) e SWU(3). Focou-se apenas nas

diferenças temporais menores do que 48 horas, visto que as falhas investigadas nesse trabalho tiveram mais ocorrências de eventos imediatamente antes e/ou imediatamente depois dentro deste intervalo de tempo.

A Figura 5.4 apresenta os histogramas para as conjunções com o subtipo AIE. Os histogramas para as demais conjunções analisadas estão apresentados nas Figuras 3.A, 4.A, 5.A e 6.A, no Apêndice.

Em todos os histogramas, o eixo x representa as diferenças temporais e o eixo y representa o número de falhas imediatamente antes e/ou imediatamente depois da conjunção de referência com a diferença temporal correspondente. Cada barra vertical representa a quantidade de diferenças temporais que foram observadas dentro de um dado intervalo de tempo. O intervalo de tempo de cada barra (tamanho do *Bin*) é exibido no canto inferior esquerdo de cada histograma. Por exemplo, na Figura 5.4(a), a primeira barra representa as falhas que ocorreram dentro do intervalo de 0 a 0,25 horas, e a segunda barra, as falhas que ocorreram dentro do intervalo 0,75 a 1 hora, em relação aos eventos de falha da conjunção *Error Code*=0x80070643 e subtipo do SWU=AIE. Portanto, nesse exemplo, o valor do *Bin* é de 0,25 horas. Também, é possível observar que algumas barras apresentaram o *Bin* igual a zero. Nesses casos, tem-se que as falhas imediatamente antes e/ou imediatamente depois ocorreram no mesmo tempo que o evento de falha da conjunção de referência (ex. Figura 5.4(g)).

Destaca-se que para algumas conjunções analisadas não foram apresentados os histogramas (ex. Figura 5.4(f)). Isso ocorreu quando as falhas imediatamente antes e/ou imediatamente depois das conjunções de referência não se ajustaram ao cenário considerado (SWU(1), SWU(2) ou SWU(3)).

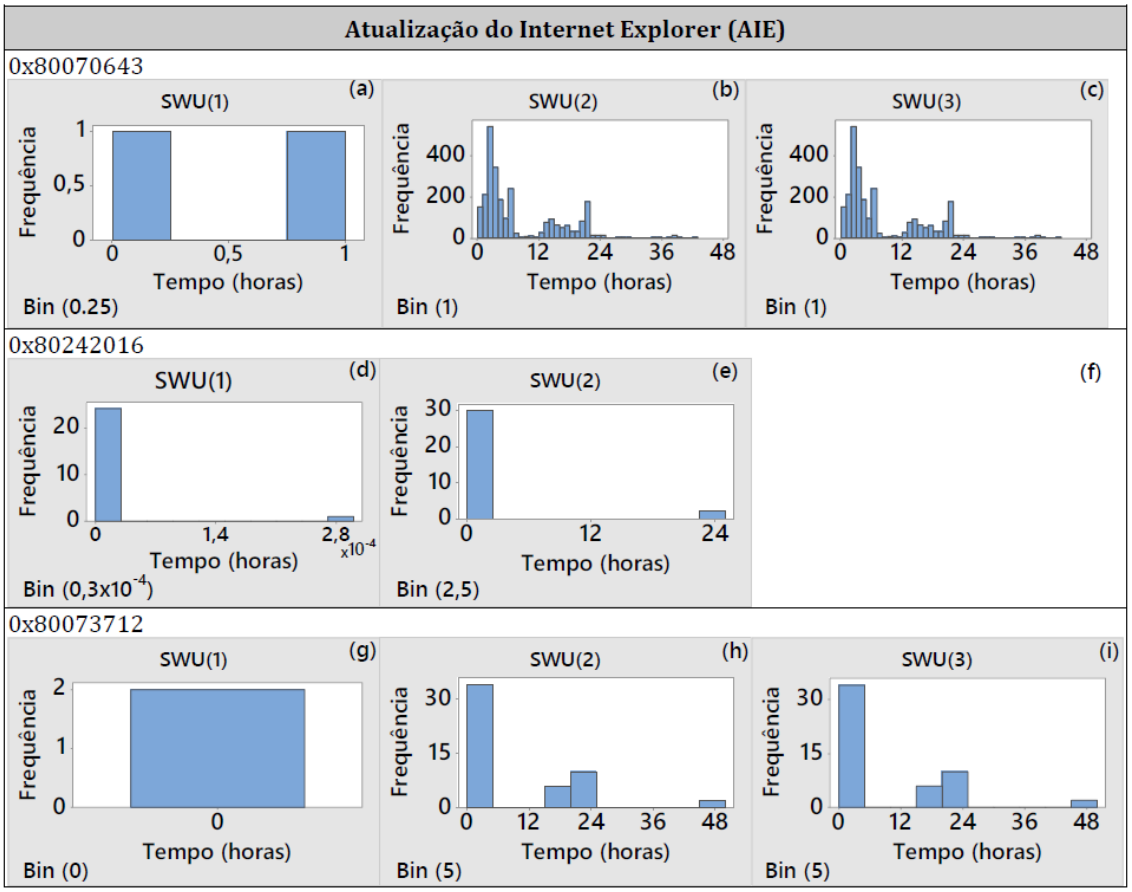


Figura 5.4. Análise temporal das conjunções do subtipo AIE.

Com base nos histogramas calculados, notou-se que as diferenças temporais das falhas que ocorreram durante o processo de instalação de uma atualização estiveram concentras dentro do intervalo de uma à quatro horas. Assim, conclui-se que o SWU tentou instalar a mesma atualização de *software* que falhou, após um determinado intervalo de tempo, e assim, essas atualizações de software que falharam durante o processo de instalação, não ocorreram repetidamente durante um curto período tempo.

Mais uma vez, observou-se que as falhas causadas pelo *Error Code* 0x80070652 não seguiram o mesmo padrão observado em outras falhas que também ocorreram durante o processo de instalação de atualizações. Isto pode ser constatado por meio das baixas diferenças de tempo entre os eventos de falha com este *Error Code* e os eventos de falha que ocorreram em seu entorno dentro do mesmo computador. Uma possível explicação para isso é que o SWU instala as atualizações de *software* sequencialmente, assim, quando ocorre uma falha causada pelo *Error Code* 0x80070652, o SWU tenta instalar outra atualização de *software* em sequência. Como ainda existe uma instalação concorrente em execução, esta nova atualização de *software* também falha com o mesmo *Error Code*.

Para os casos em que o terceiro cenário, SWU(3), apresentou valor zero de porcentagem (ex. Figura 2(f)), observou-se, que nos outros cenários da mesma conjunção (ex. histogramas (d) e (e) da Figura 5.4), as diferenças temporais são muito pequenas entre a ocorrência da conjunção de referência e as falhas em seu entorno. Este achado corrobora a evidência de que o SWU falhou como um todo e, consequentemente, causou a falha de diferentes atualizações de *software* ao mesmo tempo.

5.3.3 Tipos Mais Frequentes de Falhas das Categorias Aplicação de SO e *Kernel*

Nas Seções 5.2.1 e 5.2.3, realizou-se uma análise mais detalhada das falhas da categoria SO_{SVC}, dada a prevalência dessas falhas no conjunto de dados investigado. No entanto, como discutido na Seção 3.6.1, algumas falhas das categorias SO_{APP} e *Kernel* também tiveram considerável incidência na amostra de trabalho. Portanto, os três tipos mais frequentes de falhas dessas duas categorias são analisados nesta seção. A Tabela 5.9 lista a quantidade de eventos de falha e suas porcentagens com relação à quantidade total de falhas dessas categorias.

Tabela 5.9. Os três tipos de falha mais frequentes das categorias SO_{APP} e SO_{KNL}

Aplicação de SO (SO_{APP})		
Tipos (Total = 23)	**Nº de Eventos(Total = 1,215)**	**% de Eventos de Falha**
explorer.exe	586	48,23%
mscorsvw.exe	280	23,05%
rundll32.exe	181	14,90%
***Kernel* (SO_{KNL})**		
Tipos (Total = 35)	**Nº de Eventos(Total = 1,544)**	**% de Eventos de Falha**
VIDEO_TDR_ERROR	898	58,16%
Windows-StartupRepair	237	15,35%
DRIVER_POWER_STATE_FAILURE	123	7,97%

Falhas do programa explorer.exe são as falhas da categoria SO_{APP} com a maior ocorrência na amostra de trabalho. Em segundo lugar estão as falhas do programa mscorsvw.exe, que é um programa de geração de imagens (executáveis) do .Net Framework [De Smet 2013]. Esse programa pré-compila blocos de código de aplicações do .Net Framework [Microsoft 2016s]. Em terceiro lugar na categoria SO_{APP} estão as falhas de rundll32.exe, que é um programa que permite carregar e executar funções exportadas de bibliotecas dinâmicas no Windows, também denominadas de Dlls [Microsoft 2016t].

Na categoria SO_{KNL}, o tipo de falha de SO mais observado na amostra de trabalho foi o VIDEO_TDR_ERROR. Essa falha ocorre quando o controlador do dispositivo de vídeo detecta que já não pode mais controlar a unidade de processamento gráfico (GPU). Essa falha ocorre, frequentemente, durante a tentativa de recarregar o *driver* de vídeo [Russinovich *et al.* 2009]. O segundo tipo de falha de *Kernel* mais recorrente foi o Windows-StartupRepair, o qual indica a ocorrência de uma reinicialização do sistema para um ponto anterior de restauração causada por alterações não especificadas na configuração do sistema operacional [Microsoft 2016u]. O terceiro tipo mais recorrente de falha do *Kernel* foi o DRIVER_POWER_STATE_FAILURE, que ocorre quando um *device driver* ou uma rotina do *Kernel* falha ao completar uma operação de E/S de gerenciamento de energia dentro do intervalo de tempo padrão de 10 minutos, entrando em um estado de energia inconsistente ou inválido [Russinovich *et al.* 2009].

Similar à análise realizada para as falhas mais recorrentes da categoria SO_{SVC} (ver Seções 4.5.2), as falhas das categorias SO_{APP} e *Kernel* foram investigadas com o propósito de compreender como essas falhas estão distribuídas em relação aos diferentes grupos e computadores da amostra de trabalho.

A Tabela 5.10 apresenta a distribuição dos eventos de falha contendo os tipos mais frequentes das categorias SO_{APP} e *Kernel* listados na Tabela 5.9. Os valores entre parênteses representam o número de computadores no grupo em que a falha foi observada.

Tabela 5.10. Distribuição dos três tipos de falha mais frequentes das categorias SO_{APP} e SO_{KNL} por grupo e quantidade de computadores

Aplicação de SO (SO_{APP})					
Tipos	**G1**	**G2**	**G3**	**G4**	**Total**
explorer.exe	35 (26)	226 (102)	29(19)	296 (58)	586 (205)
mscorsvw.exe	-	115 (9)	6 (1)	159 (7)	280 (17)
rundll32.exe	-	19(10)	7 (3)	155 (11)	181 (24)
***Kernel* (SO_{KNL})**					
Tipos	**G1**	**G2**	**G3**	**G4**	**Total**
VIDEO_TDR _ERROR	825 (207)	62 (12)	-	11 (7)	898 (226)
Windows-StartupRepair	66 (53)	135 (80)	2 (2)	34 (25)	237 (160)
DRIVER_POWER_STATE_FAILURE	-	22 (10)	23 (2)	78 (16)	123 (28)

Nota-se que as ocorrências de falha relacionadas com o explorer.exe e o Windows-StartupRepair são bastante consistentes, uma vez que ocorreram em vários computadores de todos os grupos. Como já citado anteriormente, o uso intensivo do explorer.exe pode ter sido um fator para explicar sua elevada taxa de

falhas, dada a sua maior exposição e quantidade de interações com outras aplicações e componentes do sistema.

Embora tenha se observado que o tipo de falha VIDEO_TDR_ERROR teve diversas ocorrências em múltiplos computadores, suas falhas se concentraram no grupo G1. Isto sugere que influências ambientais presentes neste grupo contribuíram para provocar o elevado número de ocorrências dessa falha. Salienta-se que esse tipo de falha está relacionado com a placa gráfica do computador, sendo causadas principalmente por problemas de compatibilidade de *device drivers*, configurações de gerenciamento de energia conflitantes, mau funcionamento de *slots* PCIe e superaquecimento ou defeito na GPU, ou seja, predominantemente fatores ambientais.

Complementarmente, foram analisadas as possíveis correlações entre essas falhas das categorias SO_{APP} e *Kernel* com maior incidência na amostra de trabalho e as falhas que ocorreram imediatamente antes e/ou imediatamente depois delas. Diferentemente do método usado nas Seções 5.2.1 e 5.2.2, esta análise considerou como falhas de referência os eventos de falha das categorias SO_{APP} e *Kernel* com os tipos mais frequentes da amostra de trabalho, em vez de conjunções de *Error Codes* e Subtipos do SWU. Os resultados estão apresentados na Tabela 5.11.

As colunas 2, 3, 4 e 5 listam, respectivamente, as porcentagens de falhas imediatamente antes e/ou imediatamente depois das falhas de referência que pertencem a uma das três categorias de falha consideradas (SO_{KNL}, SO_{APP} e SO_{SVC}) e a porcentagem dessas falhas que possuem o mesmo tipo que a falha de referência (coluna MT).

Tabela 5.11. Eventos de falha imediatamente antes e/ou imediatamente depois das falhas de SO_{APP} e SO_{KNL} analisadas

Aplicação de SO (SO_{APP})				
Tipos (Total = 23)	**SO_{KNL}**	**SO_{APP}**	**SO_{SVC}**	**MT**
explorer.exe	14,69%	65,51%	19,80%	60,61%
mscorsvw.exe	0,73%	95,63%	3,64%	94,72%
rundll32.exe	4,58%	85,67%	9,74%	79,66%
Todos	10,02%	76,76%	13,22%	63,90%
***Kernel* (SO_{KNL})**				
Tipos (Total = 35)	**SO_{KNL}**	**SO_{APP}**	**SO_{SVC}**	**MT**
VIDEO_TDR_ERROR	91,51%	4,38%	4,11%	85,62%
Windows-StartupRepair	60,80%	19,44%	19,75%	27,16%
DRIVER_POWER_STATE_FAILURE	74,56%	11,84%	13,60%	71,93%
Todos	81,23%	8,73%	10,04%	71,91%

Observa-se na Tabela 5.11, que a maioria dos eventos de falha imediatamente antes e/ou imediatamente depois possuem a mesma categoria das falhas de referência. Entretanto, não possuem a mesma prevalência observada para as falhas de SO_{SVC}, em que quase todos os eventos imediatamente antes e/ou imediatamente depois das conjunções de referência também eram da categoria SO_{SVC} (ver Tabela 4.13 na Seção 4.5.2). A mesma prevalência foi observada apenas para os tipos mscorsvw.exe (SO_{APP}, 95,63%), rundll32.exe (SO_{APP}, 85,67%) e VIDEO_TDR_ERROR (SO_{KNL}, 91,51%).

Com base na coluna MT da Tabela 5.11, verifica-se que a maioria das falhas imediatamente antes e/ou imediatamente depois são do mesmo tipo da falha de referência. Isso indica um possível padrão de repetição. Este comportamento é menos presente para o tipo Windows-StartupRepair, visto que este tipo de falha de SO reinicia o sistema operacional para um ponto anterior de restauração (*checkpoint*), assim, espera-se que seja pequena a chance da mesma falha ocorrer novamente em um curto intervalo de tempo.

Com o propósito de detectar e compreender o padrão de repetição mencionado anteriormente realizou-se uma análise temporal dos três tipos mais frequentes de falha das categorias SO_{KNL} e SO_{APP}. A Tabela 5.12 apresenta este resultado, contendo a média, mediana e o desvio padrão das diferenças temporais entre as falhas imediatamente antes e/ou imediatamente depois das falhas de referência.

Tabela 5.12. Resumo da análise temporal das falhas mais frequentes das categorias SO_{APP} e SO_{KNL}

Aplicação de SO (SO_{APP})						
	Qualquer tipo/subtipo			**Mesmo Tipo**		
Tipos (Total = 23)	**Média**	**Mediana**	**Desvio Padrão**	**Média**	**Mediana**	**Desvio Padrão**
explorer.exe	572,29	101,54	1119,95	387,51	27,24	904,60
mscorsvw.exe	38,61	0	345,45	0,60	0	9,29
rundll32.exe	135,07	31,70	461,51	86,17	30,26	438,55
Todos	340,49	25,41	864,64	181,94	6,60	627,93
***Kernel* (SO_{KNL})**						
	Qualquer tipo/subtipo			**Mesmo Tipo**		
Tipos (Total = 35)	**Média**	**Mediana**	**Desvio Padrão**	**Média**	**Mediana**	**Desvio Padrão**
VIDEO_TDR_ERROR	325,33	168,26	433,79	295,38	167,45	370,01
Windows-StartupRepair	1043,92	433,91	1466,63	1335,79	643,74	1416,44
DRIVER_POWER_STATE_FAILURE	324,06	165,57	540,97	214,47	138,75	322,12
Todos	439,53	167,93	786,37	340,44	167,08	543,90

* *todos os valores em horas*.

Os resultados da Tabela 5.12 indicam que as diferenças temporais entre as falhas imediatamente antes e/ou imediatamente depois dos eventos de falha dos três tipos mais frequentes de falha das categorias SO_{KNL} e SO_{APP} são maiores em comparação com as diferenças temporais obtidas para as falhas de SO_{SVC}. Apenas para os eventos de falha com o tipo mscorsvw.exe as diferenças foram consideradas pequenas, especialmente quando as falhas imediatamente antes e/ou imediatamente depois possuem o mesmo tipo que a falha de referência, que representou 94,72% das falhas no entorno dos eventos de falha de mscorsvw.exe. Com base no campo *Message*, verificou-se que 87,07% dessas falhas, além de possuírem o mesmo tipo, apresentam o mesmo módulo defeituoso (*faulting module*) que as falhas de referência do mscorsvw.exe analisadas. Este padrão indica que a mesma falha ocorreu repetidamente em um curto período de tempo.

O *faulting module* mais observado nas falhas de mscorsvw.exe foi o bitguard.dll, presente em 54,64% de todas as falhas de mscorsvw.exe. O Bitguard é um *software* de terceiros, voltado para segurança, desenvolvido para evitar alterações indesejadas em configurações do navegador *web*. Com o propósito de

implementar seu recurso de bloqueio, a biblioteca bitguard.dll é carregada para cada aplicação em execução, uma vez que essa biblioteca fica registrada nas configurações AppInit_DLLs no registro do Windows [Microsoft 2016v].

As Figura 5.5 e 7.A (ver Apêndice) mostram os histogramas computados para a análise temporal dos três tipos mais frequentes de falha das categorias SO_{KNL} e SO_{APP}. Diferentemente do que foi observado para as falhas da categoria SO_{SVC}, em geral, as distribuições das diferenças temporais obtidas para as falhas das categorias SO_{KNL} e SO_{APP} apresentaram a propriedade de cauda longa (à direita), especialmente para as falhas da categoria SO_{KNL} (Figura 5.5). Esse é um achado importante para suportar trabalhos de modelagem de confiabilidade em sistemas operacionais. Estudos analíticos ou de simulação podem usar esta informação para selecionar a distribuição com melhor ajuste para modelar as ocorrências desses tipos de falha de SO.

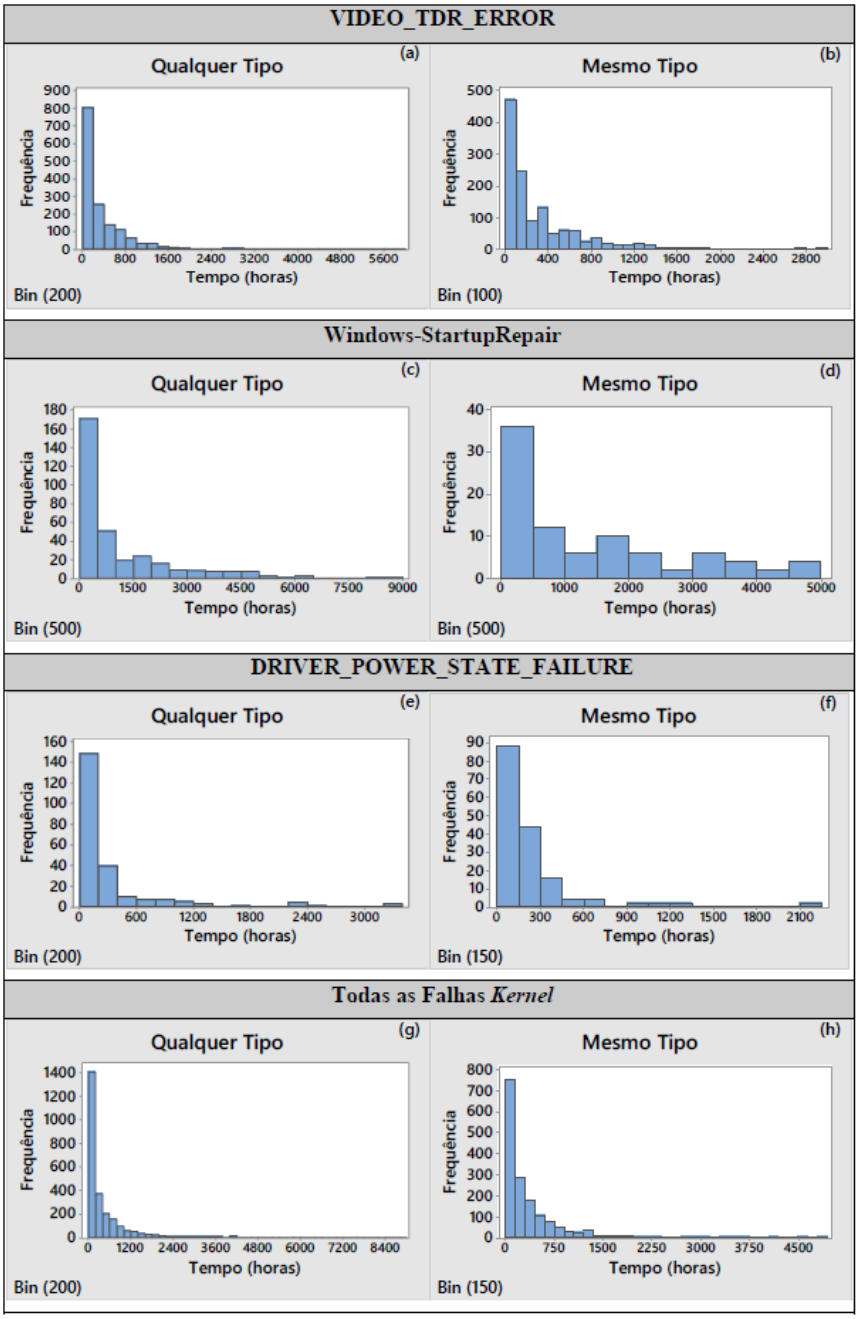


Figura 5.5. Análise temporal das falhas de SO_{KNL}.

Nas categorias SO_{KNL} e SO_{APP}, as diferenças temporais entre as falhas do mesmo tipo apresentaram menor variabilidade do que entre as falhas de diferentes tipos. Isto indica que os processos subjacentes que causam os mesmos tipos de falha são mais consistentes, em comparação com as causas de falha de diferentes tipos.

6. CONCLUSÕES DA PESQUISA

6.1 Síntese dos Resultados

Este trabalho investigou a existência de padrões entre as ocorrências de falha do sistema operacional Microsoft Windows 7. Foi necessário desenvolver um protocolo de detecção e caracterização de padrões dessas falhas, o qual foi aplicado em registros de falha coletados de quatro ambientes reais de trabalho com diferentes perfis de utilização.

Os resultados quantitativos indicam que as falhas de *Kernel* corresponderam a apenas 17,34% de todas as falhas de SO da amostra de trabalho analisada. As falhas de SO mais observadas na amostra pertencem à categoria Serviço de SO (60,63%), principalmente falhas de atualização *software*. Dessa forma, os estudos de confiabilidade de sistemas operacionais não devem se concentrar apenas em falhas de *Kernel*, como tem sido a norma na literatura (ex. [Ganapathi *et al.* 2006], [Li *et al.* 2008] e [Swift *et al.* 2003]).

Os principais achados empíricos mostram fortes evidências de correlação entre as falhas de SO. Esta é uma propriedade estatística que é muitas vezes negligenciada por estudos nesta área [Goseva-Popstojanova e Trivedi 1999], [Goseva-Popstojanova e Trivedi 2000a], [Goseva-Popstojanova e Trivedi 2000c], [Nikora e Lyu 1996], os quais assumem que falhas de *software* ocorrem de forma independente. O estudo apresentado nesta dissertação apresenta evidências empíricas de que esta suposição não é necessariamente verdadeira para as falhas de sistema operacional.

Analisando as causas das falhas mais frequentes na amostra de trabalho, identificaram-se também padrões de causalidade entre diferentes falhas de SO. Esses padrões foram observados com maior prevalência entre falhas da mesma categoria.

Especificamente sobre as falhas de Serviço de SO, que é a categoria que apresentou maior porcentagem de falhas na amostra de trabalho, verificou-se que o serviço Windows *Update* (SWU) foi o tipo de falha de SO mais predominante, correspondendo a 87,97% das falhas dessa categoria. Isso significa que as atualizações de *software* possuem um grande impacto na confiabilidade do sistema operacional.

Posteriormente, descobriu-se que quatro tipos específicos de atualização (AIE, AW7, ANF e AMO) foram os responsáveis pela maioria das falhas do SWU (88,98%). Ao investigar as causas dessas falhas de atualização, identificou-se que elas ocorreram, principalmente, devido ao mau funcionamento do *software* .Net Framework. Esse foi um, entre vários exemplos de causalidade entre as falhas analisadas. Com base nessa análise, sugere-se que uma forma de reduzir, significativamente, a taxa de falhas no sistema operacional investigado (Win7) é

realizar manutenções preventivas do .Net Framework, dado o impacto que suas falhas têm sobre demais serviços do sistema. Essa manutenção pode ser feita por meio de ferramentas voltadas à esse propósito [Microsoft 2016x]. Este é um exemplo genuíno de causalidade entre falhas de *software*, uma vez que falhas no *software* .Net Framework causam falhas no serviço Windows *Update*.

Com relação às falhas de *Kernel*, os resultados obtidos empiricamente neste estudo corroboram e são consistentes com os resultados reportados em outros estudos (ex. [Swift *et al.* 2003], [Chou *et al.* 2001] e [Murphy 2004]), ao observar que os *device drivers* foram a principal causa de falhas de *Kernel* na amostra de trabalho, em especial os *device drivers* de gerenciamento de energia e vídeo.

Com respeito às falhas relacionadas com aplicações de SO (SO_{APP}), os resultados indicam a predominância em três programas. O explorer.exe é o mais prevalente, seguido pelo mscorsvw.exe e rundll32.exe. O primeiro é amplamente utilizado, uma vez que desempenha um papel central em termos de gerenciamento de janelas e arquivos, assim, falhas neste programa tem um grande impacto sobre a experiência do usuário no sistema operacional analisado. O segundo pré-compila blocos de código de aplicações do .Net Framework, portanto, uma falha nesse programa possui um impacto maior nessas outras aplicações. Já o terceiro, permite carregar e executar funções exportadas de bibliotecas dinâmicas no Windows, as chamadas Dlls. Dessa forma, a falha desse programa também possui maior impacto em diferentes aplicações que necessitam dessas funções encontradas nas Dlls.

Com base nos *rankings* criados pelo protocolo proposto neste trabalho, observou-se que a maioria dos padrões de falhas candidatos (**PFC**s), principalmente aqueles localizados nas primeiras posições dos *rankings*, ocorreram de forma consistente em diferentes grupos de computadores para os diferentes intervalos de pesquisa analisados (2, 4 e 8 horas). Dada a diversidade dos grupos avaliados neste estudo, tal evidência reforça a hipótese de que as combinações de eventos de falha de SO detectadas nas Tabelas 5.2, 5.3 e 5.4 são padrões de falhas genuínos (**PF**s). No total, foram encontrados 45 padrões de falhas com 153.511 ocorrências.

Também, observou-se que em todas as combinações de eventos de falha de SO que compuseram os **PFC**s na Tabela 5.4, a qual exibe os *rankings* da configuração **PFC**=**EFA**→**FR**→**EFP**, continham a mesma falha de referência (**FR**), o subtipo de falha do SWU Atualização do Windows (AW7). Este subtipo também compôs **PFC**s nos *rankings* presentes nas outras tabelas (Tabelas 5.2 e 5.3). Em todas as composições de eventos analisadas, esse subtipo formou **PFC**s com outros subtipos do SWU. Esses resultados sugerem que as falhas de AW7 possuem uma forte relação com outras falhas de atualização de *software*.

Além disso, analisando os *rankings*, verificou-se que todos os subtipos de falha do SWU formaram **PFC**s com outros ou os mesmos subtipos de falha do SWU. Isso indica que falhas no serviço Windows *Update* podem causar a falha de todas as atualizações sendo realizadas por este serviço ou podem causar a falha da mesma atualização de *software* repetidas vezes, criando uma correlação entre os subtipos de falha do SWU.

Quando as causas de falha do subtipo Atualização do Windows foram analisadas por meio de seus *Error Codes*, observou-se que a maioria dessas falhas está intimamente relacionada com fatores que o SWU depende (ex. espaço em disco e o serviço RPC). O gerenciamento desses fatores, com o propósito de

controlar sua disponibilidade durante uma atualização de *software*, poderia ser uma opção para melhorar a confiabilidade do serviço Windows *Update*.

O protocolo proposto originalmente para detecção e caracterização de padrões de falhas de SO foi também adaptado para investigar as relações causais entre as ocorrências dessas falhas. Essa adaptação, descrita na Seção 4.5, visou identificar as falhas no entorno (imediatamente antes e/ou imediatamente depois) das falhas mais relevantes da amostra de trabalho. Os resultados obtidos mostram que a maioria das falhas ocorreu juntamente com outras falhas do mesmo tipo/subtipo, demonstrando um claro padrão de correlação.

Dado que a análise mencionada anteriormente não é suficiente para confirmar a presença de relações causais entre essas falhas, realizou-se uma análise temporal (ver Seções 5.2.2 e 5.2.3) das falhas, o que possibilitou identificar padrões de causalidade entre diferentes falhas de SO. Por exemplo, foram encontrados padrões de falhas que se repetiram após um determinado intervalo de tempo (geralmente entre 1 e 4 horas) ou durante um período muito curto de tempo (perto de zero segundo), e também houve casos em que diferentes falhas ocorreram ao mesmo tempo, como consequência de uma causa raiz comum.

6.2 Dificuldades Encontradas e Limitações da Pesquisa

Devido ao tamanho do conjunto de dados original (mais de 30.000 registros de falha de SO), a maior dificuldade encontrada no início da pesquisa foi elaborar um método computacionalmente eficiente para analisar essas falhas. Para contornar essa dificuldade, os registros de falha foram adicionados a um banco de dados (MySQL), possibilitando realizar diversas operações com menor custo computacional.

Outra dificuldade encontrada foi a falta de informação na literatura sobre padrões de falhas de SO. Como não foram encontrados trabalhos nessa área, foi necessário criar uma taxonomia e um protocolo próprios para analisar esse fenômeno.

Uma limitação da pesquisa é o fato de o estudo de caso ter sido realizado com apenas um sistema operacional (Microsoft Windows 7). Ressalta-se que essa limitação está circunscrita aos resultados quantitativos, pois a taxonomia e o protocolo propostos são genéricos de forma suficiente para serem aplicados na detecção e caracterização de padrões de falhas em outros sistemas operacionais.

Outra limitação da pesquisa é a amostra de trabalho. Todos os dados de falha de SO foram coletados a partir de registros realizados pelo RAC, que é uma infraestrutura desenvolvida pela Microsoft, em nível de *Kernel,* especializada na captura e armazenamento de dados de confiabilidade do Windows. Portanto, a abordagem de coleta de dados adotada nesse trabalho depende da existência do serviço RAC. O RAC é um componente que existe desde a versão Windows Vista e tem sido mantido em todas as versões futuras dos sistemas operacionais da família Windows, inclusive, a mais recente Windows 10.

6.3 Contribuição para a Literatura

Os resultados deste trabalho foram a base para a escrita de três artigos. O primeiro, intitulado "An Empirical Study on Failure Causes in a Commercial Off-the-Shelf Operating System", foi publicado nos anais da edição 2015 do Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC 2015). Este artigo ganhou o prêmio de "Best Paper" na categoria Sistemas Operacionais desse simpósio. Neste, apresentou-se a análise quantitativa e qualitativa dos registros de falha de SO, comtemplando a análise de causalidade entre as falhas do serviço Windows Update.

O segundo artigo, uma extensão do primeiro, foi submetido para o periódico ACM SIGOPS Operating System Review e se encontra no prelo. Este além de incluir a análise temporal das falhas do serviço Windows Update, também apresentou a análise de causalidade dos tipos mais frequentes de falhas das categorias Aplicação de SO e *Kernel*.

O terceiro artigo, em fase final de preparação, será submetido para uma revista internacional no primeiro semestre de 2016. Nele se descreve em detalhes o protocolo desenvolvido para detecção e caracterização de padrões de falhas de SO e os principais resultados quantitativos obtidos com a sua aplicação na amostra de trabalho.

6.4 Trabalhos Futuros

Primeiramente, como trabalho futuro, pretende-se acrescentar novos dados de falha do sistema operacional Microsoft Windows 7 com o propósito de verificar se a consistência observada nos resultados deste estudo será mantida.

Posteriormente, deseja-se aumentar a variedade da amostra de trabalho, incluindo falhas de outras versões da família Windows, assim como de diferentes sistemas operacionais (ex. Linux).

Portanto, para que seja possível realizar a análise proposta neste estudo, planeja-se realizar uma investigação de componentes em diferentes sistemas operacionais que possuam as mesmas funcionalidades do RAC, ou seja, que coletem dados de falha de SO tanto no espaço do usuário, quanto no espaço de *Kernel*. Além disso, o componente deve armazenar informações suficientes sobre essas falhas de SO que possibilitem a aplicação do protocolo apresentado.

Além disso, pretende-se integrar o intervalo de pesquisa na análise de causalidade. Portanto, será utilizada uma delimitação temporal antes e/ou depois, assim como foi feito no protocolo de detecção e caracterização de padrões de falhas, para analisar as falhas antes e/ou depois de determinadas falhas. Porém, neste caso, serão analisados os padrões de causalidade das falhas.

REFERÊNCIAS BIBLIOGRÁFICAS

[ABB 2016] ABB. “Power Generation”. http://new.abb.com/power-generation, Acessado em Janeiro 2016.

[Abbott 2012] Abbott, D. (2012). “Linux for Embedded and Real-time Applications”. Newnes, 3ª ed., 296 páginas.

[ANSI/IEEE 1991] ANSI/IEEE (1991). “Standard Glossary of Software Engineering Terminology”. STD-729-1991, ANSI/IEEE.

[Anson *et al.* 2012] Anson, S., Bunting, S. e Pearson, S. (2012). “Mastering Windows Network Forensics and Investigation”. Sybex, 696 páginas.

[Antunes e Matias 2014] Antunes, M. P. e Matias, R. (2014). “Confiabilidade de Sistemas Operacionais de Propósito Geral: Um Estudo de Caso”. Proc. 4th Brazilian Symposium on Computing Systems Engineering, pp. 2(1)-2(6).

[Avižienis *et al.* 2004] Avižienis, A., Laprie, J.-C., Randell, B. e Landwehr C. (2004). "Basic Concepts and Taxonomy of Dependable and Secure Computing," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11-33.

[Beauvisage 2009] Beauvisage, T. (2009). “Computer Usage in Daily Life”. Proc. of the International Conference on Human Factors in Computing Systems, pp.575–584.

[Bird *et al.* 2014] Bird, C., Ranganath, V.P., Zimmermann, T., Nagappan, N. e Zeller, A. (2014). “Extrinsic Influence Factors in Software *Reliability*: A Study of 200,000 Windows Machines”. Proc. 36th International Conference on Software Engineering, pp. 205-214.

[Boyce 2009] Boyce, J. (2009). “Windows 7 Bible”. Wiley Publishing, 3ª ed., 1248 páginas.

[Bruzzone et al. 2006] Bruzzone, G., Caccia, M., Bertone, A. e Ravera, G. (2006). “Standard Linux for Embedded Real-Time Robotics and Manufacturing Control Systems”. Proc. 14th Mediterranean Conference on Control and Automation, pp. 01-06.

[Candea *et al.* 2003] Candea, G., Delgado, M., Chen, M., e Fox, A. (2003). “Automatic failure-path inference: A generic introspection technique for internet applications”. Proc. 3rd IEEE Workshop on Internet Applications (WIAPP), pp.132-141.

[Chou *et al.* 2001] Chou, A., Yang, J., Chelf, B., Hallem, S. e Engler, D. (2001). "An Empirical Study of Operating Systems Errors". Proc. 18th ACM Symposium on Operating Systems Principles, pp.73-88.

[Conover 1999] Conover, W.J. (1999). "Practical Nonparametric Statistics". John Wiley & Sons, 3rd ed., 548 páginas.

[Cullen e Frey 1999] Cullen, C. e Frey, H. C. (1999). "Probabilistic Techniques in Exposure Assessment: A Handbook for Dealing with Variability and Uncertainty in Models and Inputs". Springer. 336 páginas.

[Cyganek 2013] Cyganek, B. (2013). "Object Detection and Recognition in Digital Images: Theory and Practice". Wiley, 1ª ed., 548 páginas.

[Dai *et al.* 2005] Dai, Y. S., Xie, M. e Poh, K. L. (2005). "Modeling and Analysis of Correlated Software Failures of Multiple Types". IEEE Transactions on Reliability (Volume: 54, Issue: 1), pp.100-106.

[De Smet 2013] De Smet, B. (2013). "C# 5.0 Unleashed", 1ª ed., Sams Publishing, 1700 páginas.

[Dijkstra 1975] Dijkstra, E. (1975). "Selected Writings on Computing: A personal perspective". How Do We Tell Truths That Might Hurt? EWD498.

[Endler 2015] Endler, M. "Windows 7 Dominates Desktop, XP Share Slips". Disponível em: <http://www.informationweek.com/software/operating-systems/windows-7-dominates-desktop-xp-share-slips/d/d-id/1235016>. Acessado em Dezembro 2015.

[Ganapathi and Patterson 2005] Ganapathi, A. e Patterson, D. (2005). "Crash Data Collection: A Windows Case Study". Proc. International Conference on Dependable Systems and Networks, pp. 280-285.

[Ganapathi *et al.* 2006] Ganapathi, A., Ganapathi, V., e Patterson, D. (2006). "Windows XP *Kernel* Crash Analysis". Proc. 20th Conference on Large Installation System Administration, pp. 149-159.

[Goseva-Popstojanova e Trivedi 1999] Goseva-Popstojanova, K. e Trivedi, K.S. (1999). "The Effects of Failure Correlation on Software *Reliability* and Performability", Fast Abstracts 29th Int'l Symp. Fault Tolerant Computing, pp. 45-46.

[Goseva-Popstojanova e Trivedi 2000a] Goseva-Popstojanova, K. e Trivedi, K.S. (2000). "Effects of Failure Correlation on Software in Operation". Proc. Pacific Rim International Symposium on Dependable Computing (PRDC2000), pp. 69-76.

[Goseva-Popstojanova e Trivedi 2000b] Goseva-Popstojanova, K. e Trivedi, K.S. (2000). "Architecture Based Software *Reliability*", Proc. Int'l Conf. Applied Stochastic System Modeling, pp. 41-52.

[Goseva-Popstojanova e Trivedi 2000c] Goseva-Popstojanova, K. e Trivedi, K.S. (2000). "Failure Correlation in Software *Reliability* Models", IEEE Trans. On *Reliability*, Vol.49, No.1, pp. 37-48.

[HPDCS 2016a] Microsoft. "DATASET CHARACTERIZATION". Disponível em: <http://hpdcs.facom.ufu.br/osr-team/index.php>. Acessado em Janeiro 2016.

[HPDCS 2016b] HPDCS. "Operating System Reliability Analysis Tool". Disponível em: <http://hpdcs.facom.ufu.br/osrat>. Acessado em Janeiro 2016.

[Kalyanakrishnam *et al.* 1999] Kalyanakrishnam, M., Kalbarczyk, Z. e Iyer, R. (1999). "Failure Data Analysis of a LAN of Windows NT Based Computers," Proc. 18th IEEE Symposium on Reliable Distributed Systems, pp. 178-187.

[Lee *et al.* 2011] Lee, T., Mitschke, K., Schill, M. E. e Tanasovski, T. (2011). "Windows PowerShell 2.0 Bible", 1st ed., John Wiley & Sons, Hoboken, 696 páginas.

[Leveson e Turner 1993] Leveson, N. G. e Turner, C. S. (1993). "An Investigation of the Therac-25 Accidents". IEEE Computer , vol. 26 , no. 7 , pp.18-41.

[Lewis 1995] Lewis, E. E. (1995). "Introduction to *Reliability* Engineering". Wiley Publishing, 2ª ed., 464 páginas.

[Li *et al.* 2008] Li, P.L., Ni, M., Xue, S., Mullally, J.P., Garzia, M. e Khambatti, M. (2008). "*Reliability* Assessment of Mass-Market Software: Insights from Windows Vista". Proc. 19th Int'l Symp. on Software *Reliability* Engineering, pp. 265-270.

[Lipow 1982] Lipow, M. (1982). "Number of Faults per Line of Code". IEEE Transactions on Software Engineering, pp. 437-43.

[Lyu 2007] Lyu, M. (2007). "Software *Reliability* Engineering: A Roadmap," Proc. 29th Int. Conf. Softw. Eng., Future Softw. Eng., pp.153 -170.

[Matias *et al.* 2013] Matias R., Oliveira G. e Araújo L. (2013). "Operating System *Reliability* from the Quality of Experience Viewpoint: An Exploratory Study". Proc. 28th ACM Symposium on Applied Computing, pp.1644-1649.

[Matias *et al.* 2014] Matias, R., Prince, M., Borges, L., Sousa, C. e Henrique, L. (2014). "An Empirical Exploratory Study on Operating System *Reliability*". Proc. 29th ACM Symposium on Applied Computing, pp. 1523-1528.

[Microsoft 2016a] Microsoft. "*Reliability* Analysis Component". Disponível em: <http://technet.microsoft.com/en-us/library/cc774636(v=ws.10).aspx>. Acessado em Janeiro 2016.

[Microsoft 2016b] Microsoft. "*Reliability* Infrastructure". Disponível em: <https://technet.microsoft.com/pt-br/library/cc732712(v=ws.10).aspx>. Acessado em Janeiro 2016.

[Microsoft 2016c] Microsoft. “How to Use *Reliability* Monitor”. Disponível em: <http://windows.microsoft.com/en-us/windows7/how-to-use-*reliability*-monitor>. Acessado em Janeiro 2016.

[Microsoft 2016d] Microsoft. “Understanding the System Stability Index”. Disponível em: <https://technet.microsoft.com/en-us/library/cc749032(v=ws.10).aspx>. Acessado em Janeiro 2016.

[Microsoft 2016e] Microsoft. “Win32_*Reliability*Records class”. Disponível em: <https://msdn.microsoft.com/en-us/library/windows/desktop/ee706630%28v =vs.85%29.aspx?f=255&MSPPError=-2147217396>. Acessado em Janeiro 2016.

[Microsoft 2016f] Microsoft. “What are *Error Codes* and How Can I Use Them?”. Disponível em: <http://windows.microsoft.com/en-us/windows/what-are-error-codes#1TC=windows-7>. Acessado em Janeiro 2016.

[Microsoft 2016g] Microsoft. “Microsoft *web site*”. Disponível em: <https://www.microsoft.com/en-us/windows>. Acessado em Janeiro 2016.

[Microsoft 2016h] Microsoft. “Microsoft TechNet”. Disponível em: https://technet.microsoft.com/en-us/>. Acessado em Janeiro 2016.

[Microsoft 2016i] Microsoft. “Microsoft Community”. Disponível em: <http://answers.microsoft.com/en-us?auth=1>. Acessado em Janeiro 2016.

[Microsoft 2016j] Microsoft. “Windows *Update* Showing Error 80070643”. Disponível em: <http://answers.microsoft.com/en-us/insider/forum/insider_wintp-insider_*update*/windows-*update*-showing-error-80070643/0e53bf0f-8843-45a1-b3c4-0940c516c8d9>. Acessado em Janeiro 2016.

[Microsoft 2016k] Microsoft. “Windows *Update* Error 80070643”. Disponível em: <http://windows.microsoft.com/en-us/windows/windows-*update*-error-80070643#1TC=windows-7>. Acessado em Janeiro 2016.

Microsoft 2016l] Microsoft. “Troubleshooting Issues When You Use the Discovery Wizard to Install an Agent”. Disponível em: <https://technet.microsoft.com/en-us/library/ff358634.aspx?f=255&MSPPError=-2147217396>. Acessado em Janeiro 2016.

[Microsoft 2016m] Microsoft. “Windows *Update* Error 800B0100”. Diponível em: <http://windows.microsoft.com/en-us/windows/windows-*update*-error-800b0100#1TC=windows-7>. Acessado em Janeiro 2016.

[Microsoft 2016n] Microsoft. “Windows *Update* Agent - *Error Codes*”. Disponível em: <http://social.technet.microsoft.com/wiki/contents/articles/15260.windows-*update*-agent-error-codes.aspx>. Acessado em Janeiro 2016.

[Microsoft 2016o] Microsoft. “Windows *Update* Error 80242016”. Disponível em: <http://windows.microsoft.com/en-us/windows7/windows-*update*-error-80242016>. Acessado em Janeiro 2016.

[Microsoft 2016p] Microsoft. “Windows *Update* Error 0x80073712”. Disponível em: http://windows.microsoft.com/en-us/windows-8/windows-*update*-error-0x80073712>. Acessado em Janeiro 2016.

[Microsoft 2016q] Microsoft. “*Error Code*: 80070652 (Can't Install KB2540162)”. Disponível em: <http://answers.microsoft.com/en-us/windows /forum/all/error-code-80070652-cant-install-kb2540162/1689922e-eab9-4b10-b44e-2c110d6c3bfa>. Acessado em Janeiro 2016.

[Microsoft 2016r] Microsoft. “Windows *Error Code* 0x800f0902”, Disponível em: <https://social.technet.microsoft.com/Forums/windowsserver/en-US/1eb8c5c2-64bf-4d07-8020-dac902230688/addwindowsfeature-sometimes-fails-with-error-0x800f0902-the-operation-cannot-be-completed-because?forum=winservermanager>. Acessado em Janeiro 2016.

[Microsoft 2016s] Microsoft. “Assemblies in the Common Language Runtime”. Disponível em <https://msdn.microsoft.com/en-us/library/hk5f40ct(v=vs.90).aspx>. Acessado em Janeiro 2016.

[Microsoft 2016t] Microsoft. “Dynamic-Link Libraries”. Disponível em: <http://msdn.microsoft.com/en-us/library/ms682589.aspx>. Acessado em Janeiro 2016.

[Microsoft 2016u] Microsoft. “Did System Uptime Error Cause Chkdisk?” Disponível em: <http://answers.microsoft.com/en-us/windows/forum/all/did-system-uptime-error-cause-chkdisk/8d55525c-9f5b-4278-bd87-1598bce009ee>. Acessado em Janeiro 2016.

[Microsoft 2016v] Microsoft. “What is BitGuard?”. Disponível em: <http://answers.microsoft.com/en-us/windows/forum/windows_xp-pictures/what-is-bitguard/f3c7c34f-dca1-4441-8eed-39c197e1b2d5?auth=1>. Acessado em Janeiro 2016.

[Microsoft 2016x] Microsoft. "Microsoft .NET Framework Repair Tool". Disponível em: <https://www.microsoft.com/en-us/download/details.aspx?id=30135>. Acessado em Janeiro 2016.

[Mood *et al.* 1974] Mood, A.M., Graybill, F.A. and Boes, D. C. (1974). “Introduction to the Theory of Statistics”. McGraw-Hill, 3rd ed., 480 páginas.

[Murphy 2004] Murphy, B. (2004). “Automating Software Failure Reporting”, Magazine Queue - System Failures, pp. 42-48.

[Murphy 2008] Murphy, B. (2008). “*Reliability* Estimates for the Windows Operating System”. Microsoft Research Cambridge. Disponível em: http://www.dcl.hpi.uni-potsdam.de/meetings/mshpsummit/slides/brendan.murphy.pdf>. Acessado em Janeiro 2016.

[NetMarketShare 2015] NetMarketShare. “Desktop Operating System Market Share”. Disponível em: <http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0>. Acessado em Dezembro 2015.

[Nikora e Lyu 1996] Nikora, A.P. e Lyu M.R. (1996) “Software *Reliability* Measurement Experience”, Handbook of Software *Reliability* Engineering, M.R.Lyu (Ed.), McGraw-Hill, pp. 255-301.

[Rayner 2009] Rayner, J. C. W., Thas O. e Best, D. J. (2009) “Smooth Tests of Goodness of Fit: Using R”. John Wiley & Sons, 2nd ed., 304 páginas.

[Russinovich *et al.* 2009] Russinovich, M., Solomon, D. A. and Ionescu, A. 2009. ‘Microsoft Windows Internals’. Microsoft Press, 6ª ed., 2ª parte, 672 páginas.

[Salfner *et al.* 2006] Salfner, F., Schieschke, M. e Malek, M. (2006). “Predicting Failures of Computer Systems: A Case Study for a Telecommunication System”. Proc. 20th international conference on Parallel and distributed processing, pp.348-348.

[Shelly *et al.* 2010] Shelly, G., Freund, S. e Enger, R. (2010). “Microsoft Windows 7: Complete”. Course Technology, 1ª ed., 488 páginas.

[Shields e Rangarjan 2013] Shields, P. and Rangarjan, N. (2013). “A Playbook for Research Methods: Integrating Conceptual Frameworks and Project Management”. New Forums Press, 292 páginas.

[StatCounter Global Stats 2016] StatCounter Global Stats. “Top Desktop Operating Systems Worldwide”. Disponível em: <http://gs.statcounter.com/#os-ww-monthly-201411-201601>. Acessado em Janeiro 2016.

[Swift *et al.* 2003] Swift, M. M., Bershad, B. N. e Levy, H. M. (2003). “Improving the *Reliability* of Commodity Operating Systems”. Proc. 19th ACM Symposium on Operating Systems Principles, pp. 207-222.

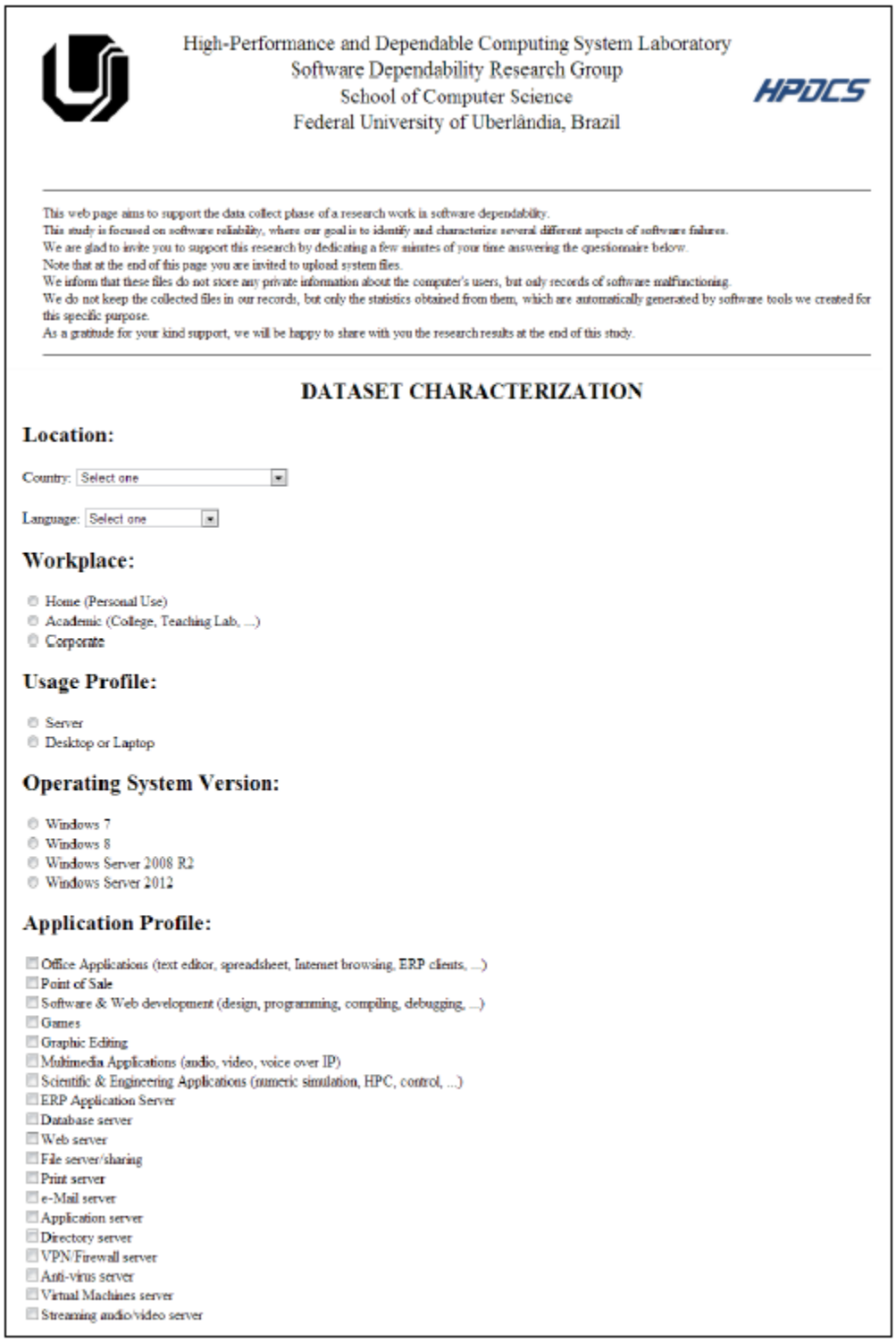
[Technet 2016] Technet. “Understanding Component-Based Servicing”. Disponível em: <http://blogs.technet.com/b/askperf/archive/2008/04/23/ understanding-component-based-servicing.aspx>. Acessado em Janeiro 2016.

[Thomas 2012] Thomas, O. (2012). “MCTS 70-680 Rapid Review: Configuring Windows 7”. Microsoft Press, 1ª ed., 228 páginas.

[Trivedi 2001] Trivedi, K.S. (2001). “Probability and Statistics with *Reliability*, Queueing, and Computer Science Applications”. Wiley-Interscience, 2nd ed., 830 páginas.

[Xu *et al.* 1999] Xu, J., Kalbarczyk, Z. e Iyer, R. (1999). “Networked Windows NT System Field Failure Data Analysis”. Proc. Pacific Rim International Symposium on Dependable Computing, pp. 178-185.

APÊNDICE



High-Performance and Dependable Computing System Laboratory
Software Dependability Research Group
School of Computer Science
Federal University of Uberlândia, Brazil

HPDCS

This web page aims to support the data collect phase of a research work in software dependability.
This study is focused on software reliability, where our goal is to identify and characterize several different aspects of software failures.
We are glad to invite you to support this research by dedicating a few minutes of your time answering the questionnaire below.
Note that at the end of this page you are invited to upload system files.
We inform that these files do not store any private information about the computer's users, but only records of software malfunctioning.
We do not keep the collected files in our records, but only the statistics obtained from them, which are automatically generated by software tools we created for this specific purpose.
As a gratitude for your kind support, we will be happy to share with you the research results at the end of this study.

DATASET CHARACTERIZATION

Location:

Country: Select one

Language: Select one

Workplace:

- Home (Personal Use)
- Academic (College, Teaching Lab, ...)
- Corporate

Usage Profile:

- Server
- Desktop or Laptop

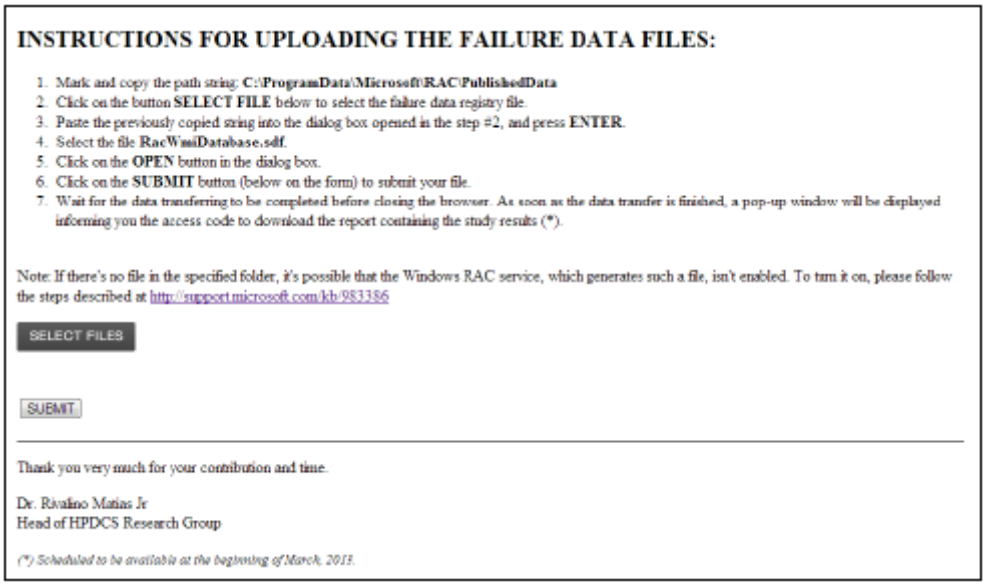
Operating System Version:

- Windows 7
- Windows 8
- Windows Server 2008 R2
- Windows Server 2012

Application Profile:

- Office Applications (text editor, spreadsheet, Internet browsing, ERP clients, ...)
- Point of Sale
- Software & Web development (design, programming, compiling, debugging, ...)
- Games
- Graphic Editing
- Multimedia Applications (audio, video, voice over IP)
- Scientific & Engineering Applications (numeric simulation, HPC, control, ...)
- ERP Application Server
- Database server
- Web server
- File server/sharing
- Print server
- e-Mail server
- Application server
- Directory server
- VPN/Firewall server
- Anti-virus server
- Virtual Machines server
- Streaming audio/video server

Figura 1.A. Formulário *online* para caracterização dos computadores pesquisados.



INSTRUCTIONS FOR UPLOADING THE FAILURE DATA FILES:

1. Mark and copy the path string: **C:\ProgramData\Microsoft\RAC\PublishedData**
2. Click on the button **SELECT FILE** below to select the failure data registry file.
3. Paste the previously copied string into the dialog box opened in the step #2, and press **ENTER**.
4. Select the file **RacWmiDatabase.sdf**.
5. Click on the **OPEN** button in the dialog box.
6. Click on the **SUBMIT** button (below on the form) to submit your file.
7. Wait for the data transferring to be completed before closing the browser. As soon as the data transfer is finished, a pop-up window will be displayed informing you the access code to download the report containing the study results (*).

Note: If there's no file in the specified folder, it's possible that the Windows RAC service, which generates such a file, isn't enabled. To turn it on, please follow the steps described at http://support.microsoft.com/kb/983386

SELECT FILES

SUBMIT

Thank you very much for your contribution and time.

Dr. Rivalino Matias Jr
Head of HPDCS Research Group

(*) Scheduled to be available at the beginning of March, 2013.

Figura 2.A. Instruções para envio do arquivo SDF por meio do formulário *online*.

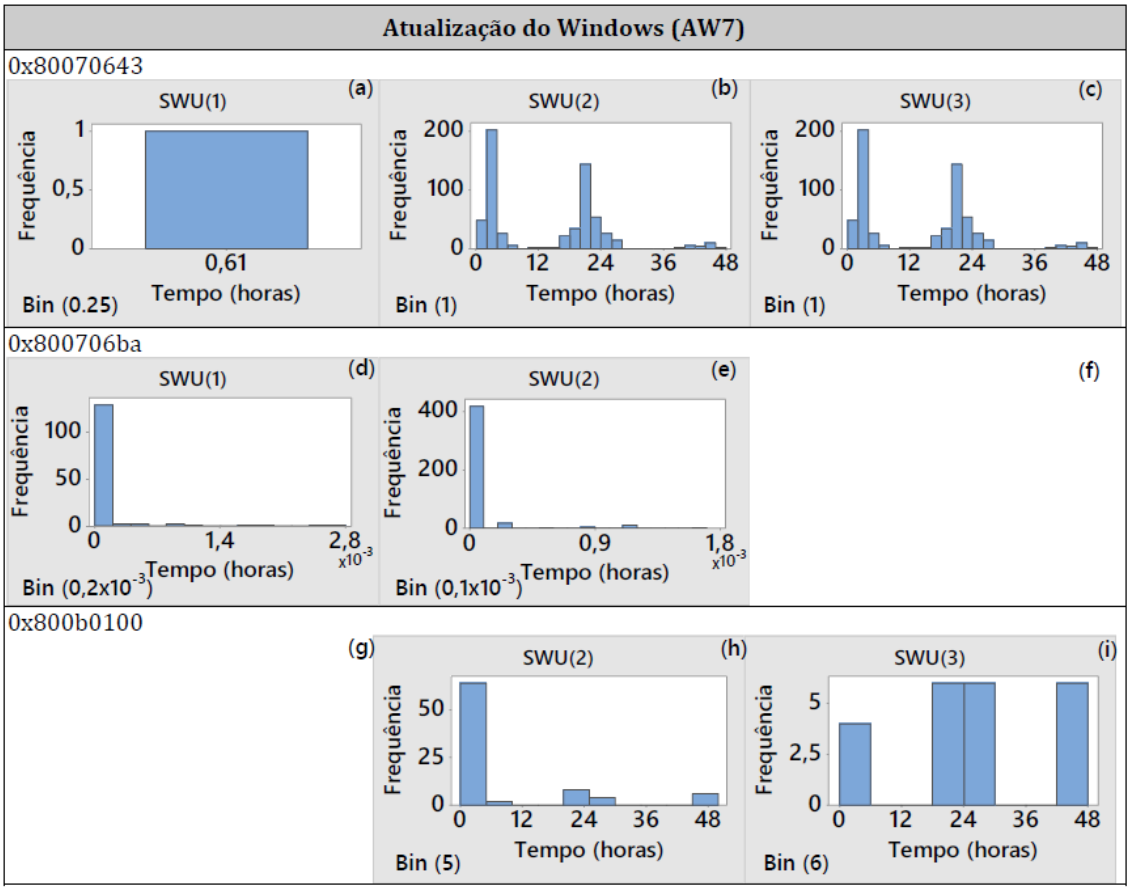


Figura 3.A. Análise temporal das conjunções do subtipo AW7.

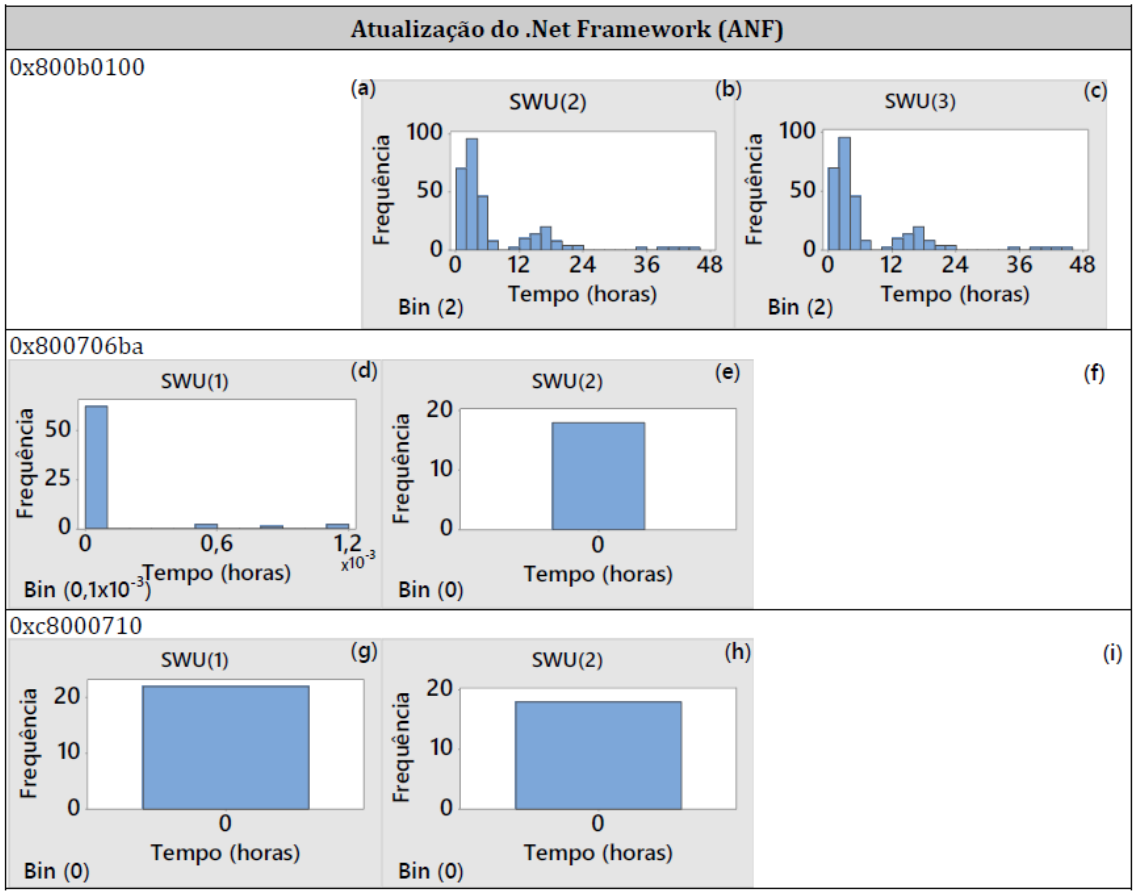


Figura 4.A. Análise temporal das conjunções do subtipo AFN.

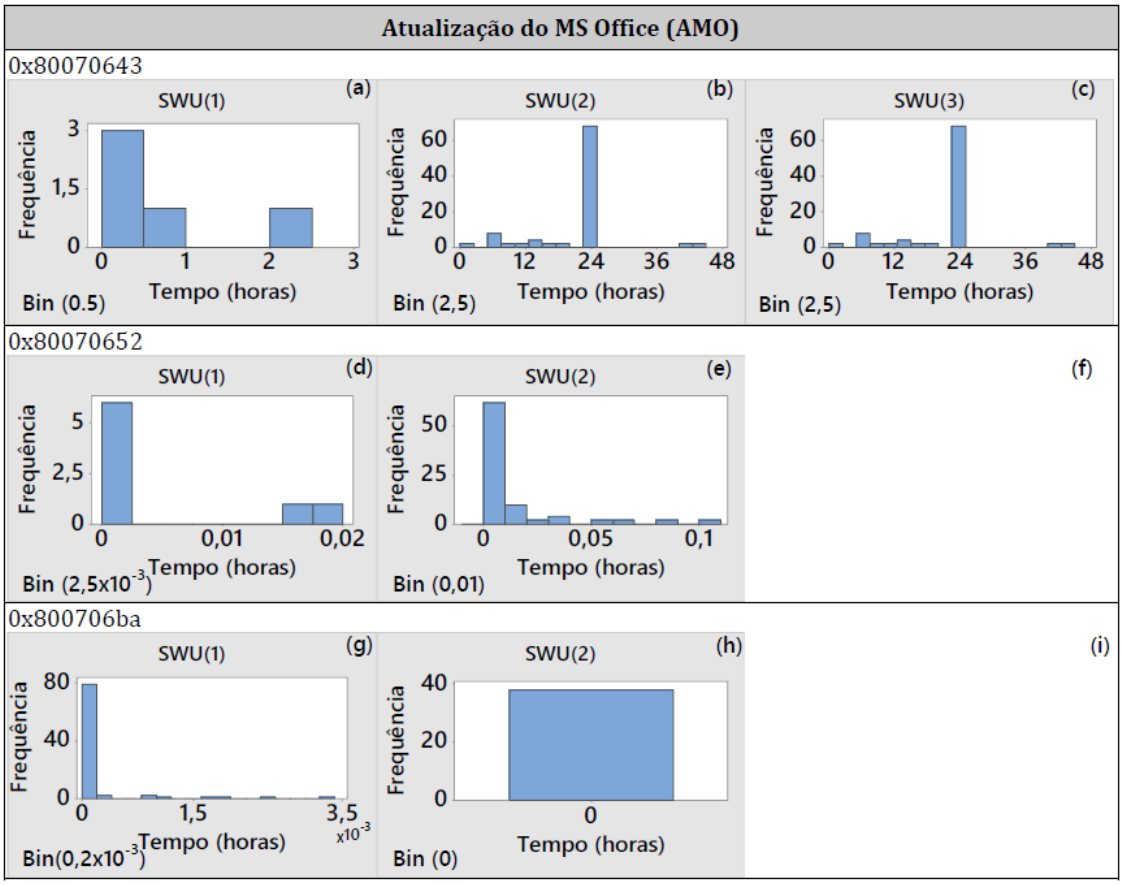


Figura 5.A. Análise temporal das conjunções do subtipo AMO.

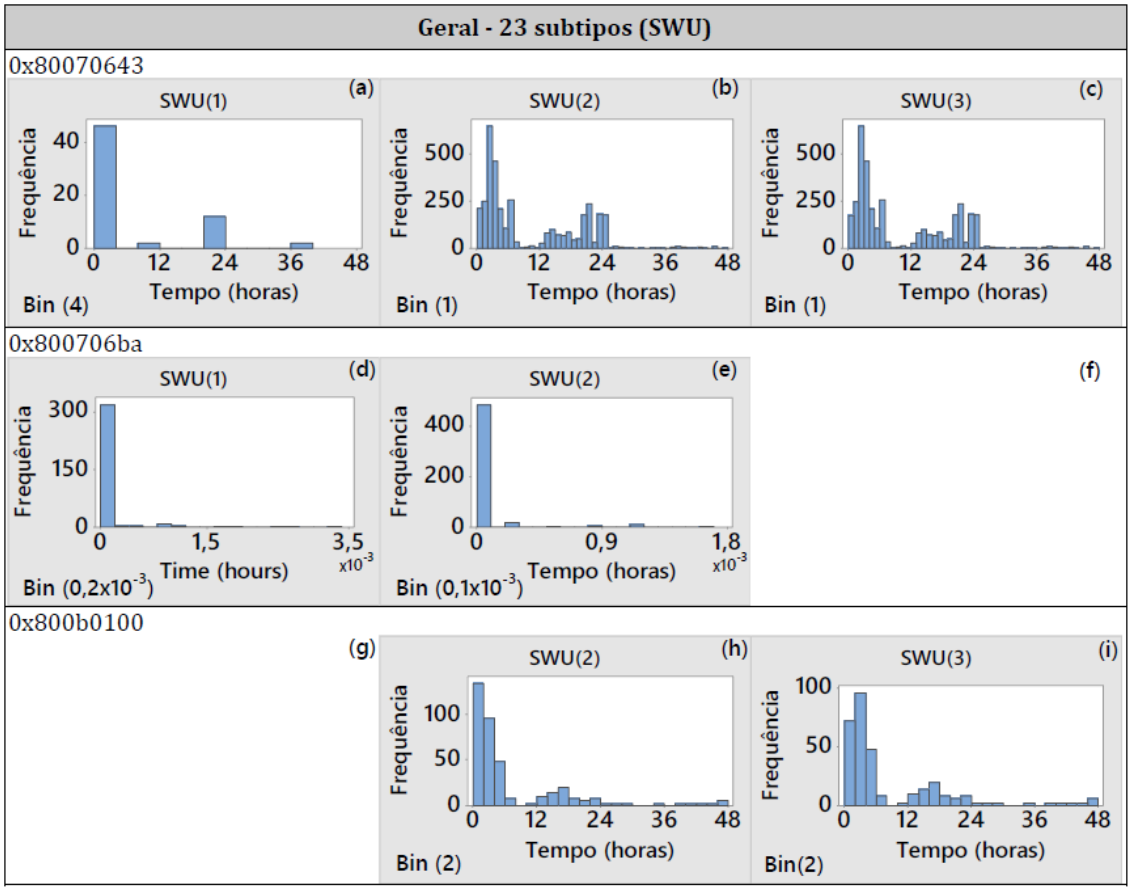


Figura 6.A. Análise temporal das conjunções do tipo SWU.

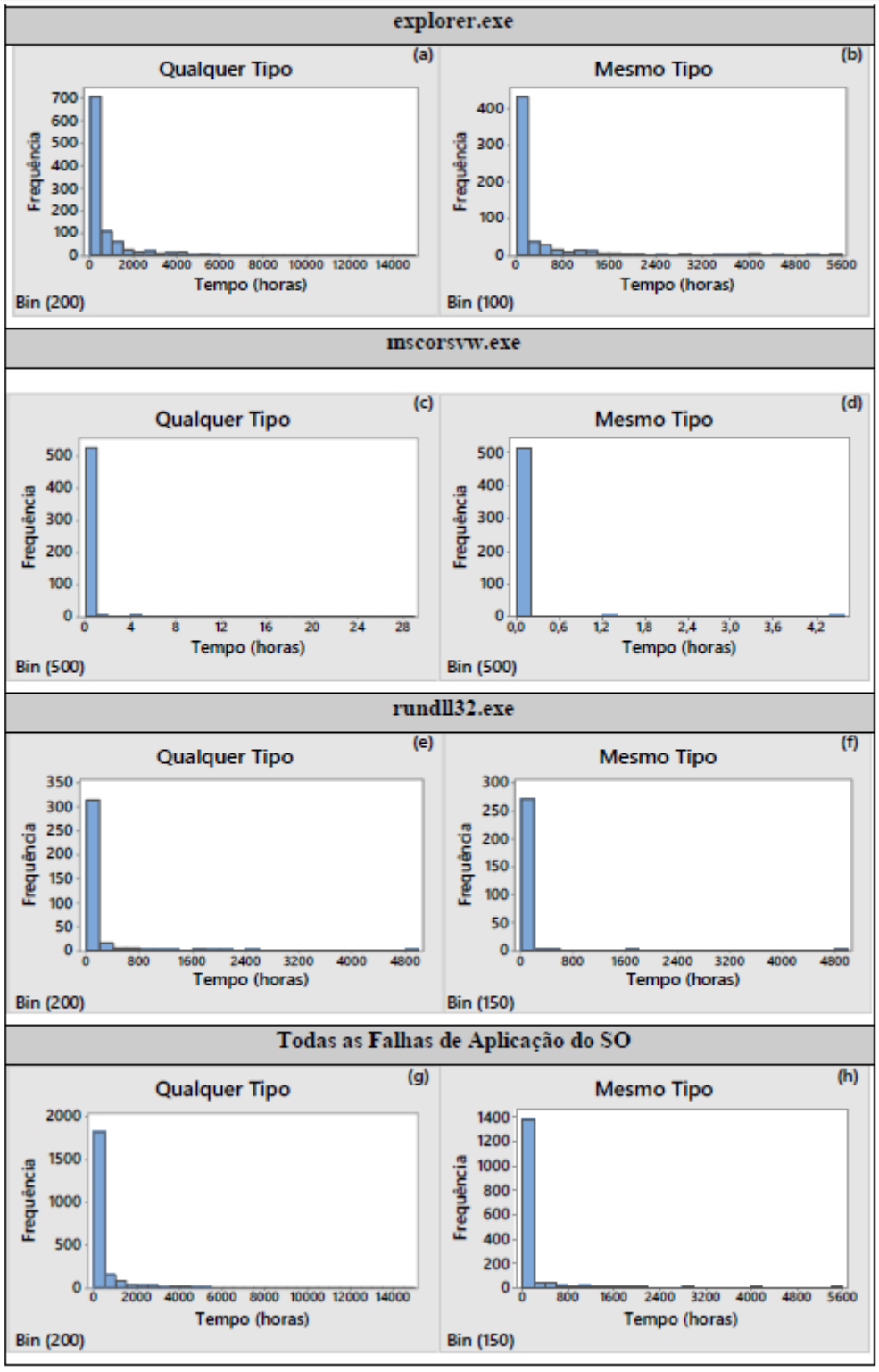


Figura 7.A. Análise temporal das falhas de SO_{APP}.