Dezembro 2019
ISSN 1413-9014
E-ISSN 2317-2193

# CADERNOS DO IME
## Série Informática
## Volume 43

# CADERNOS DO IME
# Série Informática
# Volume 43

Publicação do Instituto de Matemática e Estatística da Universidade do Estado do Rio de Janeiro. Periodicidade semestral, com circulação em junho e dezembro.



**Normalização, divulgação e distribuição:**
Biblioteca do Centro de Ciências de Tecnologia A (CTC/A) da rede Sirius de Biblioteca da UERJ – ctca@uerj.br

**Organização e Edição do Volume 43:**
Vera Maria B. Werneck
Maria Clicia Stelling de Castro

**Correspondência:**
Universidade do Estado do Rio de Janeiro
Instituto de Matemática e Estatística
Editores do Cadernos do IME - Série Informática
Rua São Francisco Xavier, 524 - Pavilhão Reitor João Lyra Filho,
6º andar, sala 6019 B
Maracanã - 20550-900 – Rio de Janeiro, RJ

**Telefax:** +55 21 2334-0144
**e-mail:** cadernos_inf @ime.uerj.br
**site:** http://www.e-publicacoes.uerj.br/index.php/cadinf/



**Produção e editoração gráfica:**
Vera Maria B. Werneck
Maria Clicia Stelling de Castro

Organizadores:
**Vera Maria B. Werneck**
**Maria Clicia Stelling de Castro**

Contato: **cadernos_inf@ime.uej.br**

**Dezembro 2019**
**ISSN 1413-9014**
**E-ISSN 2317-2193**

# CADERNOS DO IME
# Série Informática
# Volume 43

# Nota dos Editores

Este é o segundo volume lançado em 2019 e é uma edição onde publicamos 2 Artigos submetidos à Revista e os três melhores artigos do ERAD-RJ 2019 - Escola Regional de Alto Desempenho realizada no período de 04 a 06 de Setembro no CEFET/RJ. Após o período com problemas estruturais relacionados à manutenção física da revista,), impactando a capacidade da revista em manter o fluxo de publicações no Cadernos do IME Série Informática. Agradecemos aos nossos revisores e à Comissão Geral e de ERAD-RJ 2019. Temos convicção de que, bons ventos virão a nosso favor, o que se demonstra na qualidade dos artigos da presente edição.

Dezembro de 2019



**Cadernos do IME – Série Informática**
**ISSN 1413-9014**
**E-ISSN 2317-2193**

# Apresentação

Para o Volume 43 dos Cadernos do IME – Série Informática aceitamos dois artigos submetidos e convidamos os três melhores artigos da Escola Regional de Alto Desempenho (ERAD-RJ) de 2019 para publicarem seus melhores trabalhos. Os artigos do ERAD-RJ foram revisados por pelos menos três revisores da comunidade de Alto Desempenho do Rio de Janeiro.

O primeiro artigo, *NGAV (Next-Generation Antivirus) Especialista na Detecção de Cyber-Ataques* aborda os aplicativos malware que podem causar grandes prejuízos. Uma de suas características principais é a renovação sistemática, o que dificulta a sua detecção.

O segundo artigo, *Modelando com o Método ERi*c, um Aplicativo para Gerenciamento da Fila de um Restaurante* mostra como foram aplicadas todas as seis etapas que compõem o método ERi*c (Engenharia de Requisitos Intencional) no estudo de caso relacionado a gerência de uma fila de espera num restaurante. O método ERi*c é uma colaboração à Engenharia de Requisitos Orientada a Metas (GORE). Uma importante particularidade do método é sua capacidade de reduzir a complexidade dos modelos iStar (i*), utilizados para a criação de modelos intencionais.

O terceiro artigo, *Sistema de voto eletrônico utilizando a blockchain* aborda a implementação de um sistema de voto eletrônico usando a técnica de *Blockchain*, para garantir a segurança da votação, e um banco de dados descentralizado e criptografado. Além da segurança, o sistema proposto permite a realização de auditoria em cada fase do processo de eleição, desde o código fonte até a base de dados, e provê transparência na apuração.

O quarto artigo, *NOPL-Erlang: Programação multicore transparente em linguagem de alto nível*, apresenta os avanços da linguagem NOPL e de seu compilador específico, além da sua integração com a arquitetura Erlang. A combinação da NOPL com o compilador proporciona um ambiente de programação em alto nível que expressa execução concorrente de forma transparente. Os resultados mostram redução no tempo de execução à medida em que se aumenta o número de núcleos disponíveis.

O quinto artigo, *Avaliando a Detecção e o Tratamento de Flash Crowds Utilizando o SimGrid,* aborda os conteúdos disponíveis na Internet que geram grande interesse e recebem uma grande quantidade de acessos rapidamente são eventos denominados flash crowds. Eles precisam ser detectados e tratados de forma rápida e eficiente. Porém, a avaliação desses eventos em ambientes reais tem um alto custo financeiro. Assim, este artigo propõe uma simulação que permite que recursos da nuvem sejam contratados dinamicamente para tratar o problema, utilizando o simulador SimGrid combinado com soluções existentes

encontradas na literatura. Desta forma, pode-se avaliar a qualidade das soluções em cenários maiores sem custos adicionais e que representem situações reais de flash crowds.

Renovamos o convite aos pesquisadores para enviarem suas contribuições para os próximos números dos Cadernos do IME – Série Informática, para mantermos o mesmo padrão de qualidade obtido na presente edição.

Dezembro de 2019

Vera M. Werneck e Maria Clicia Stelling de Castro

**Organizadores do Volume 43**

cadernos_inf@ime.uerj.br

# CADERNOS DO IME
# Série Informática

**Normas para publicação de trabalhos**

A revista Cadernos do IME - Série Informática é um veículo para divulgação de trabalhos acadêmicos, científicos ou profissionais nas áreas de informática, ciência da computação e correlatas, sob responsabilidade do Departamento de Informática e Ciência da Computação (DICC) do Instituto de Matemática e Estatística (IME) da Universidade do Estado do Rio de Janeiro (UERJ).

A revista é semestral, sendo publicada em junho e dezembro. São aceitos trabalhos inéditos e que apresentem elementos de originalidade, em português, inglês ou espanhol.

Os Cadernos do IME aceitam os seguintes tipos de contribuições:
- Artigo técnico, trabalho de pesquisa com elementos de originalidade;
- Tutorial, texto introdutório a áreas específicas da ciência da computação, com revisão de literatura e organização dos conceitos relacionados;
- Resenha / Revisão, revisão de literatura e análise crítica de produtos da área de ciência da computação e informática;
- Resumo, texto destinado à informação sobre trabalhos de graduação e de pós-graduação;
- Comunicação, texto destinado à divulgação de opiniões, pesquisas em andamento, lançamentos e divulgação de eventos.

Os artigos e tutoriais não devem ultrapassar 20 páginas e as demais formas de publicação devem se restringir a no máximo 10 páginas.

Todos os trabalhos enviados para publicação nos Cadernos do IME – Série Informática, independente de tipo de contribuição, devem seguir o modelo para publicação de artigos da SBC – Sociedade Brasileira de Computação.

Os arquivos digitais no formato fonte (.DOC) e PDF deverão ser encaminhados aos editores da série.

A submissão é eletrônica e em fluxo contínuo no endereço http://www.e-publicacoes.uerj.br/index.php/cadinf. As chamadas de trabalho para volumes especiais estão disponíveis http://www.ime.uerj.br/cadernos/cadinf/.

Os conteúdos e pontos de vista expressos nos trabalhos publicados são de inteira responsabilidade dos autores.

# *NGAV (Next-Generation Antivirus)* Especialista na Detecção de *Cyber*-Ataques

**Sidney M. L. Lima [1], Ricardo P. Pinheiro [2], Danilo M. Souza [2], Sthéfano H. M. T. Silva [2], Petrônio G. Lopes [2], Rafael D. T. de Lima [2], Jemerson R. de Oliveira [2], Thyago de A. Monteiro [2], Sérgio M. M. Fernandes [2], Edison de Q. Albuquerque [2]**

[1] Departamento de Eletrônica e Sistemas, Universidade Federal de Pernambuco – Recife, Brasil

[2] Departamento de Computação, Universidade de Pernambuco – Recife, Brasil

sidney.lima@ufpe.br,{rpp3, dms2, shtms, pgl, rdtl, jro, tam, smurilo, edison}@ecomp.poli.br

***Abstract.*** *Almost all malware running on web-server are PHP codes. The present paper creates a NGAV (Next-Generation Antivirus) expert in auditing web-based threats, specifically from PHP files, in real time. Our antivirus monitors 11,777 malicious behaviors that cyber-attacks can do when executing directly from a malicious web server to a service on a personal computer. Our NGAV achieves an average accuracy of 97.50% in distinguishing between benign and malware web scripts through data science, artificial intelligence and machine learning. Our antivirus can supply the limitations of the commercial antiviruses as for the detection of Web fileless attack.*

***Keywords****. Malware; Antivirus; Artificial Neural Networks; Real-time malware detection; Computer Forensics.*

***Resumo.*** *Quase todos os malware executados em servidores web são códigos PHP. Este trabalho cria um NGAV (Next Generation Antivirus - Antivírus de Próxima Geração) especialista em detectar ameaças web em arquivos PHP, em tempo real. Nosso antivírus monitora 11.777 comportamentos maliciosos que o cyber-ataque possa fazer quando executado diretamente de um servidor web malicioso para um serviço em um computador pessoal. O nosso NGAV alcança uma precisão média de 97.50% na distinção entre scripts web benignos e malware através de ciência dos dados, inteligência artificial e máquinas de aprendizado estatístico. O nosso antivírus tem a capacidade de suprir as limitações do antivírus comerciais quanto à detecção de cyber-ataques oriundos de servidores web.*

***Palavras-Chave.*** *Malware; Antivírus; Redes Neurais Artificiais; Detecção de Malware em tempo real; Forense computacional.*

## 1. Introdução

A internet vem se caracterizando como o principal meio de comunicação na sociedade contemporânea. A internet se notabiliza pela convergência de todos os meios de comunicação previamente existentes. Através da rede mundial de computadores, é possível assistir televisão, ouvir rádio, ler jornal e ter acesso a qualquer outro forma de transmissão de informação entre diferentes povos, idiomas e culturas. Com a popularização da internet, os estudantes criam seu próprio ambiente virtual de estudo,

proveem seu próprio conteúdo e interagem de forma ativa e constante na busca pelo conhecimento. A rede mundial de computadores impulsiona a criatividade, habilidades tecnológicas, possibilita a abertura de distintas visões, além de habilidades de comunicação e de aprendizado (MINNESOTA/USA, 2008).

Como efeito colateral, a crescente popularização da internet propicia que a produção de malware continue crescendo de forma rápida ainda durante alguns anos visto que a internet é o principal meio de propagação de aplicações maliciosas (INTEL, 2018). Apenas em 2016, foram lançados mais de 7.100.000 (sete milhões e cem mil) malware, um aumento de 47,3% em relação ao ano de 2015 (INTEL, 2018). "Malware" é uma junção dos termos "malicioso" e "software". O malware tem como principal objetivo acessar um dispositivo alheio sem permissão explícita de seu proprietário (LIMA, *et al.*, 2018)(CERT.BR, 2016). Logo, senhas bancárias, redes sociais, fotos ou vídeos íntimos podem ser furtados a partir da *cyber*-infecção por malware.

Enfatiza-se que grande parte dos transtornos provocados por malware são irreversíveis. Logo, cada vez mais se vem investindo na segurança digital através de novas tecnologias em antivírus, firewall e biometria. Estima-se que os serviços de antivírus estão presentes em 95% dos computadores pessoais, além de 84% dos internautas terem serviços de firewall e 82% possuírem atualizações automáticas ativadas no seu sistema operacional Microsoft (MICROSOFT, 2017).

Apesar da presença massiva de mecanismos de *cyber*-vigilância em praticamente todos os computadores pessoais, os ataques cibernéticos vêm causando prejuízos bilionários e em escalas cada vez maiores (MICROSOFT, 2017). Uma das razões desse insucesso é porque assim que uma vulnerabilidade é solucionada, *cyber*-criminosos surgem com outra tática (SOPHOS, 2014). Atualmente, ao invés de infecções convencionais, através de arquivos executáveis portáveis, os *cyber*-ataques modernos empregam ataques "sem arquivos" ou *fileless*. Tecnicamente, ataques "sem arquivos" são lançados diretamente de um servidor web malicioso em direção a um serviço responsivo em um computador pessoal (CONRAD, *et al.*, 2017). Em 2017, das novas vulnerabilidades observadas, apenas 24% eram originadas no computador pessoal, e as demais (76%) advinham do servidor. A redução dos ataques provenientes do cliente reflete a tendência na diminuição de táticas invasivas as quais visam atingir o cliente diretamente (SKYBOX, 2018).

Pesquisa da *Symantec* estima que as principais formas de *cyber*-infecções, pela internet, são sites comuns que foram comprometidos e infectados com código malicioso (SYMANTEC, 2012). A lista completa contendo as categorias mais perigosas de ataques a páginas web pode ser vistas na Figura 1. É interessante notar que sites de conteúdo adulto/pornográfico estão fora do top cinco, em décimo lugar. Por sua vez, os blogs, comumente com conteúdo ideológico e religioso, possuem em média o triplo de ameaças em comparação a sites adulto-pornográficos (SYMANTEC, 2012). Conclui-se que, independentemente do seu comportamento, um internauta não está a salvo quanto a infecções visto que conselhos convencionais já não são mais úteis como, por exemplo, não acessar sites pornográficos visando evitar cyber-invasões.

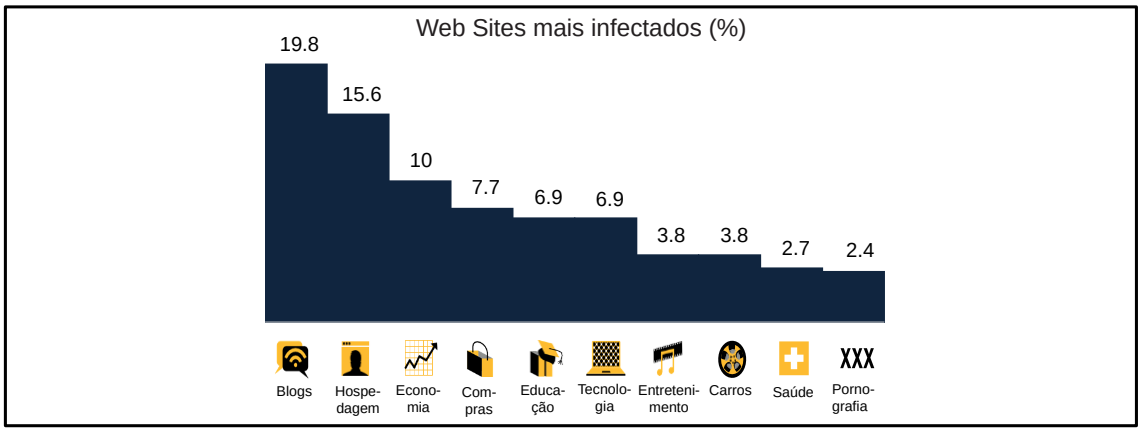



**Figura 1. Categorias mais perigosas de Web Sites, de acordo com a Symantec (SYMANTEC, 2012).**

Uma vez que os scripts maliciosos estejam instalados nos sites comprometidos, eles lançam dinamicamente ataques por meio de navegadores web para a vítima (computador pessoal). Quase todos os arquivos contendo malware executados em servidores web são códigos PHP (uma linguagem em scripts para servidores, comumente usada em sites) (SOPHOS, 2014). É crescente o uso de scripts PHP maliciosos confeccionados para que servidores disseminem atividades ilegais (SOPHOS, 2014). Como adversidade, ataques originados no servidor via malware PHP são bastante difíceis de catalogar (SOPHOS, 2014). Graças à natureza do negócio de hospedagem de sites, quando um uma empresa descobre que um de seus servidores está infectado, é bem mais vantajoso simplesmente apagar este servidor e configurar um novo ao invés de tentar descobrir o que aconteceu (SOPHOS, 2014). Como nem tais empresas nem seus parceiros de segurança sabem exatamente o que aconteceu, geralmente o novo servidor é rapidamente infectado também (SOPHOS, 2014).

Em síntese, ataques oriundos de servidores web, por meio de malware PHP, tem capacidade de ludibriar os mecanismos de *cyber*-vigilância (SOPHOS, 2014). Então, o trabalho proposto investiga (*i*) os principais antivírus comerciais quanto à perícia forense de PHP maliciosos. A detecção de malware variou entre 0% a 78,50% a depender do antivírus. Em média, houve a detecção 16,82% das pragas virtuais. Com aspecto desfavorável, os antivírus, em média, atestaram falsos negativos e foram omissos em 49,49% e 33,69% dos casos, respectivamente. Além disso, cerca de 60% dos antivírus não foram capazes de diagnosticar qualquer uma das amostras maliciosas. Enfatiza-se que em nosso estudo, os malware PHP analisados têm as suas atuações maliciosas documentadas e catalogadas em bancos de dados (VIRUSSHARE, 2019). Mesmo assim, mais da metade dos antivírus comerciais avaliados não tinham qualquer conhecimento sobre as existências dos arquivos PHP malware investigados.

Há uma gama de motivos para que o ataque “sem arquivos” represente um desafio tão relevante para os antivírus tradicionais (PALOALTO, 2013). Uma delas diz respeito à aquisição do arquivo PHP visto que seria necessária a permissão do provedor de conteúdo de onde ele foi executado remotamente. Na prática forense digital, no entanto, empresas de hospedagem costumam trabalhar de forma desintegrada e não compartilhar informações com as empresas de *cyber*-segurança (SOPHOS, 2014). Logo, as estratégias de atuação das empresas de hospedagem web atrapalham e retardam o enfrentamento a malware PHP.

Atualmente, as organizações buscam suprir as deficiências dos antivírus tradicionais através de mecanismos de *cyber*-segurança nomeados de NGAVs (*Next Generation Antivirus*). As soluções NGAVs buscam reconhecer padrão de comportamento de malware através do uso de inteligência artificial, aprendizagem de máquina e ciência de dados (SANS, 2019). A recomendação dos pesquisadores é que os NGAVs adicionem várias camadas de inteligência na detecção de *cyber*-ameaças (SKYCURE, 2016). Logo, os NGAVs do estado-da-arte propõem extrair características do arquivo, de maneira preventiva, antes de executá-lo. O executável passa por um processo de *disassembling* visando a Engenharia Reversa do arquivo suspeito. Logo, o executável pode ser estudado e, portanto, é possível investigar a intenção maliciosa do arquivo através de máquinas de aprendizado estatístico. Essa metodologia, nomeada de análise estática, é capaz de obter taxas médias de acertos superiores a 90% na detecção de aplicativos malware (LIMA, *et al.*, 2018).

A análise estática, no entanto, pode ser facilmente contornada por um ataque web "sem arquivos". Em síntese, análise estática de características é inválida mediante ataque "sem arquivos" visto que não há como um computador pessoal periciar códigos fontes armazenados e executados em um servidor web remoto. A incapacidade da análise estática em detectar ataques "sem arquivos" tem mudado o foco da pesquisa de malware para a determinação de características que possam identificar comportamento malicioso como um processo, e não pelos meios empregados na análise estática. Então, ao invés da impraticável análise estática, a extração de características do nosso NGAV diz respeito à auditória do comportamento do sistema operacional e não mais a inexequível análise do código fonte do arquivo suspeito.

De modo a validar o nosso NGAV, o trabalho proposto desenvolve um ambiente controlado nomeado *Web-Server Next Generation Sandbox*. Em nosso ambiente, são virtualizados o servidor web malicioso e o computador pessoal, respectivamente. Então, o computador pessoal (cliente) requisita a página Web suspeita do servidor virtualizado. A partir daí, os comportamentos maliciosos, oriundos do ataque "sem arquivos", são auditados por nossa *Web-Server Next Generation Sandbox*. Tais comportamentos maliciosos servem como atributos de entrada das máquinas de aprendizado estatístico.

Quantos aos cenários e experimentos, são explorados diferentes parâmetros das redes neurais artificiais empregadas como máquina de aprendizado estatístico. O nosso NGAV (*iii*) alcança um desempenho médio de 97.50% na distinção entre aplicativos PHP benignos e malware. Então, o presente artigo demonstra que a inteligência artificial é uma boa alternativa para as fabricantes dos antivírus comerciais. As limitações dos mecanismos de *cyber*-segurança tradicionais podem ser supridas por nosso NGAV dotado de ambiente controlado especialista em auditar ataques *fileless*. Ao invés de modelos baseados em listas negras, a nossa *engine* emprega ciência de dados, aprendizagem de máquina e inteligência artificial na identificação de comportamentos maliciosos.

O trabalho proposto tem, como destaque, as seguintes contribuições:

- Investigação dos principais 86 antivírus comerciais quanto à identificação de arquivos PHP malware. Em média, houve a detecção 16,82% das pragas virtuais.

- O nosso NGAV (*Next Generation Antivirus*) alcança um desempenho médio de 97.50% na distinção entre arquivos benignos e *malware*, acompanhado de um tempo de treinamento médio de apenas 0,04 segundos [1].
- O antivírus criado possibilita a detecção preventiva das ameaças virtuais, em ambiente controlado, antes de alcançarem as máquinas dos clientes.
- Criação de base de dados, nomeada PAEMAL, visando ser empregada, como *benchmark*, na verificação da qualidade dos antivírus.

## 2. Limitações dos Antivírus Comerciais

Apesar de ser questionado há mais de uma década, o *modus operandi* dos antivírus é baseado em assinaturas quando o arquivo suspeito é consultado em bases de dados nomeadas de lista negra (SANS, 2019). Isso quer dizer, o malware suspeito é comparado a uma lista negra confeccionada a partir de denúncias prévias e isso requer que alguns clientes já tenham sido infectados. Antivírus baseados em detecção de assinaturas possuem bons resultados quando se deparam com ameaças conhecidas, mas encontram grandes dificuldades no combate a aplicativos malware recém-criados (PALOALTO, 2013). Tal dificuldade é agravada em relação a códigos PHP mal-intencionados visto que navegar, assim como outras atividades web, são de tempo real por natureza (PALOALTO, 2013).

Logo, basta que o *hash* do arquivo investigado não esteja na lista negra do antivírus para que o malware não seja detectado. O *hash* funciona como um identificador único de um dado arquivo. Então, dadas as limitações dos antivírus comerciais, não é uma tarefa difícil desenvolver e distribuir variantes de uma aplicação mal intencionada. Para isso, basta fazer pequenas alterações no malware original com rotinas que, efetivamente, não tem qualquer utilidade a exemplo de laços de repetição e desvios condicionais sem instruções em seus escopos. Essas alterações sem utilidade, no entanto, tornam o *hash* do malware modificado diferente do *hash* do malware original. Consequentemente, o malware, incrementado com rotinas nulas, não será detectado pelo antivírus o qual catalogou o malware original. Cabe ressaltar a existência de ferramentas (*exploits*) responsáveis por criar e distribuir variantes, de forma automatizada, de um mesmo malware original. Conclui-se que antivírus, baseados em assinaturas, têm efetividade nula quando submetidos a variantes de um mesmo malware (SANS, 2019).

Por intermédio da plataforma VirusTotal, o trabalho proposto investiga os principais antivírus comerciais com seus respectivos resultados apresentados na Tabela 1. Os resultados dos 86 principais antivírus comerciais estão disponíveis no nosso repositório autoral (PAEMAL, 2019). Foram empregados 200 PHP maliciosos. O objetivo é verificar a quantidade de pragas virtuais catalogadas pelos antivírus. A motivação é que a aquisição de novas pragas virtuais assume papel importante no combate a aplicações mal-intencionadas. Logo, quanto maior for a base de dados de malware, nomeada de lista negra, melhor tende a ser a defesa provida pelo antivírus. A Figura 2 exibe o diagrama da metodologia proposta em diagrama de blocos. Inicialmente, os aplicativos malware são enviados ao servidor pertencente à plataforma VirusTotal

[1] Na primeira linha da Tabela 3, há a descrição técnica da melhor configuração adotada pelo nosso NGAV (*Next Generation Antivirus*).

(VIRUSTOTAL, 2019). Após isso, eles são analisados pelos antivírus comerciais vinculados ao VirusTotal. A plataforma permite a possibilidade de emissão de três tipos diferentes de diagnósticos; malware, benigno e omissão.

Quanto à primeira possibilidade do VirusTotal, o antivírus detecta a malignidade do arquivo suspeito. No ambiente experimental proposto, todos os arquivos submetidos são aplicativos malware de domínio público (VIRUS, 2019). Logo, o antivírus acerta quando detecta a malignidade do arquivo investigado. A detecção do malware indica que o antivírus provê um serviço robusto contra *cyber*-invasões. Na segunda possibilidade, o antivírus atesta a benignidade do arquivo investigado. Logo, no estudo proposto, quando o antivírus alega a benignidade do arquivo, trata-se de um caso de falso negativo visto que todas as amostras são maliciosas. Isso quer dizer, o arquivo investigado é malware, no entanto, o antivírus atesta benignidade, de forma equivocada. Na terceira possibilidade, o antivírus não emite opinião sobre o arquivo suspeito. A omissão indica que o arquivo investigado jamais foi avaliado pelo antivírus tão pouco ele possui robustez para avaliá-lo em tempo real. A omissão do diagnóstico, por parte do antivírus, aponta a sua limitação quanto a serviços em larga escala.

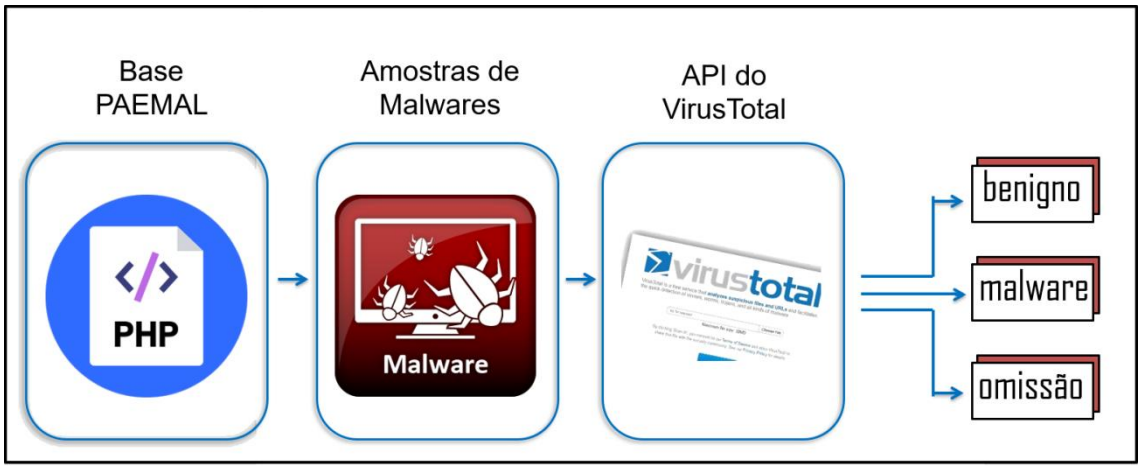



**Figura 2. Diagrama da API do VirusTotal**

A detecção dos aplicativos malware PHP variou de 0% a 78,50% a depender do antivírus avaliado. Em média, os 86 antivírus foram capazes de detectar 16,82% das pragas virtuais avaliadas, com desvio padrão de 21,88. O desvio padrão elevado indica que a detecção de arquivos maliciosos pode sofrer variações abruptas a depender do antivírus escolhido. Determina-se que a proteção, contra invasões cibernéticas, está em função da escolha de um antivírus robusto dotado de uma grande e atualizada lista negra. Em média, os antivírus atestaram falsos negativos em 49,49% dos casos, com desvio padrão de 38,32. Atestar a benignidade de um malware pode implicar em prejuízos irrecuperáveis. Uma pessoa ou instituição, por exemplo, passaria a confiar em uma determinada aplicação maliciosa quando, de fato, trata-se de um malware. Ainda como aspecto desfavorável, cerca de 57% não emitiram opinião em qualquer uma das 200 amostras maliciosas. Em média, os antivírus foram omissos em 33,68% dos casos, com desvio padrão de 45.61. A omissão do diagnóstico aponta a limitação dos antivírus quanto à detecção de malware em tempo real.

**Tabela 1. Resultado dos melhores e piores antivírus comerciais em relação à malware PHP. Resultados expandidos estão no nosso repositório autoral (PAEMAL, 2019).**

| Antivírus | Detecção (%) | Falso negativo (%) | Omissão (%) |
|---|---|---|---|
| Ikarus | **78.50** | 16.00 | 5.50 |
| GData | 59.00 | 39.50 | 1.50 |
| AegisLab | 55.50 | 42.50 | 2.00 |
| Avast | 54.50 | 45.50 | 0.00 |
| MAX | 54.50 | 42.50 | 3.00 |
| AVG | 54.00 | 46,00 | 0.00 |
| Kaspersky | 50.50 | 48.00 | 1.50 |
| ZoneAlarm | 50.50 | 48,00 | 1.50 |
| Avira | 49.00 | 50.00 | 1.00 |
| MicroWorld-eScan | 47.00 | 53.00 | 0.00 |
| BitDefender | 47.00 | 51.50 | 1.50 |
| Ad-Aware | 47.00 | 50.50 | 2.50 |
| Emsisoft | 47.00 | 53.00 | 0.00 |
| ALYac | 46.50 | 52.00 | 1.50 |
| F-Prot | 5.00 | 94.50 | 0.50 |
| TrendMicro | 4.00 | 93.00 | 3.00 |
| ClamAV | 3.50 | 94.50 | 2.00 |
| VIPRE | 3.50 | 96.00 | 0.50 |
| TotalDefense | 2.50 | 97.00 | 0.50 |
| Jiangmin | 2.50 | 96.00 | 1.50 |
| AhnLab-V3 | 2.00 | 98.00 | 0.00 |
| K7GW | 1.50 | 98.50 | 0.00 |
| K7AntiVirus | 1.50 | 98.50 | 0.00 |
| VBA32 | 1.00 | 98.50 | 0.50 |
| nProtect | 0.50 | 44.50 | 55.00 |
| ViRobot | 0.50 | 99.50 | 0.00 |
| Yandex | 0.50 | 98.50 | 1.00 |
| Panda | 0.50 | 99.50 | 0.00 |

**Tabela 2: Miscelânea de classificações providas pelos antivírus comerciais. O total de classificações está no repositório autoral (PAEMAL, 2019).**

| Antivírus | VirusShare_f0cba054906c17c94b6852c6088b47b0.php | VirusShare_13f71688e77649255460d68258f3e450.php | VirusShare_d293667cf4aad8acce2b86258462bcdb.php |
|---|---|---|---|
| Ikarus | Trojan.JS.Tadtruss | Backdoor.IRCBot.ADDS | JS:Trojan.JS.Agent.SJP |
| GData | Benign | Win.Trojan.Downloader-68 | Benign |
| AegisLab | Trojan.Script.Agent.dtkph | Benign | Benign |
| Avast | Benign | Suspicious_GEN.F47V0405 | Suspicious_GEN.F47V0614 |
| MAX | EXP/Blacole.EB.4 | malware | malware |
| AVG | JS:Decode-DB | PHP:Multicom-A | JS:Agent-DWO |
| Kaspersky | Win.Trojan.Iframe-68 | Backdoor.IRCBot.ADDS | JS:Trojan.JS.Agent.SJP |
| ZoneAlarm | Benign | Benign | Benign |
| Avira | Benign | Benign | Benign |
| MicroWorld-eScan | Benign | Benign | JS.TrojanjQuery.8C8B |
| BitDefender | Trojan.JS.Iframe.wq | Benign | HEUR:Trojan.Script.Generic |
| Ad-Aware | Benign | Benign | Benign |
| Emsisoft | Trojan.JS.IFrame.ANM | Benign | HTML/Phishing.m |
| ALYac | JS/BlacoleRef.E | JS.Redirector.AX | Benign |
| Baidu | Trojan.JS.IFrame.ANM | Benign | Benign |
| Bkav | Omission | Omission | Omission |
| McAfee-GW-Edition | HEUR_HTJS.HDJSFN | Benign | Benign |
| Arcabit | Exploit | TrojanDownloader:PHP/RunShell.A | Benign |
| McAfee | JS.Blacole.H | PHP/SillyDlScript.HFI | Benign |
| Antiy-AVL | Malware | PHP/Downloader.A | HTML/Infected.WebPage.Gen2 |
| F-Secure | TrojWare.JS.Agent.exi | Benign | TrojWare.JS.Agent.CUS |

Inclui-se como adversidade, no combate a aplicações mal intencionadas, o fato dos antivírus comerciais não possuírem um padrão na classificação dos aplicativos *malware* como visto na Tabela 2. Nós escolhemos 3 dos 998 arquivos malware PHP de modo a exemplificar a miscelânea de classificações dadas pelos antivírus comerciais. Como não existe um padrão, os antivírus dão os nomes que desejam, por exemplo, o McAfee-GW-Edition pode identificar um *malware* como "HEUR_HTJS.HDJSFN" e o McAfee, pertencente a mesma empresa, identificá-lo como "JS.Blacole.H". Logo, a falta de um padrão atrapalha as estratégias de *cyber*-vigilância visto que cada categoria de malware deve ter tratamentos (vacinas) distintos. Conclui-se que é inviável o aprendizado de máquina supervisionado visando reconhecimento de padrão de categorias de PHP malware. Devido a esse emaranhado confuso de classificação multi-classe, providas pelos

especialistas (antivírus) como visto na Tabela 2, é estatisticamente improvável que alguma técnica de aprendizado de máquina adquira capacidade de generalização.

## 3. Estado-da-Arte

O *modus operandi* dos antivírus comerciais é majoritariamente a identificação de PHP malware com bases em assinaturas. Dada as limitações dos antivírus comerciais, o estado-da-arte propõe extrair e analisar as características dos aplicativos malware através de ciência dos dados, máquinas de aprendizados estatísticos e Inteligência Artificial. A intenção do estado-da-arte é detectar o *cyber*-ataque antes mesmo dele alcançar o computador pessoal do cliente.

PEKTAS, *et al* (2017) emprega 17.900 arquivos maliciosos (Exe 32 bits, HTML, FLASH, Java e APK) do banco de dados VirusShare (VIRUSSHARE, 2019). O trabalho assegura o rótulo das amostras através da plataforma VirusTotal e usa uma técnica dinâmica para analisá-los, o que é feito através de duas ferramentas *sandboxes*: VirMon e Cuckoo. Inicialmente, os arquivos são enviados para a plataforma VirusTotal, para garantir que as amostras maliciosas estejam infectadas com códigos maliciosos. Depois disso as amostras são submetidas às ferramentas Cuckoo e VirMon, aos quais executarão os arquivos e criarão um relatório das ações executadas. Estes relatórios são analisados por diferentes algoritmos de aprendizado online (PA-I, PA-II, CW, AROW e NHERD) através da plataforma online jubatus. Nos algoritmos de aprendizagem *on-line* 5 pesos diferentes (1.0, 2.0, 3.0, 4.0 e 5.0) são utilizados, com validação cruzada de 10 k-*fold*. A sugestão é que a ferramenta Weka foi utilizada embora não haja tal relato no artigo. A obra de PEKTAS, *et al* (2017) é comparado com diferentes trabalhos contendo diferentes técnicas de classificação de arquivos maliciosos, ao qual obteve o quinto melhor resultado. Cabe ressaltar que é possível verificar que as acurácias, em média, obtiveram melhores resultados, o que indica ser uma metodologia promissora. Na obra de PEKTAS, *et al* (2017), a sensibilidade dos resultados é descrita através de uma matriz de confusão e do algoritmo CW. No melhor cenário, o trabalho de PEKTAS, *et al* (2017), atinge uma precisão de 94% no treinamento e 92,5% no teste (PEKTAS, *et al.*, 2017).

SESHAGIRI, *et al* (2016) emprega 789 Javascripts malignos oriundos da Malware Domain List e 1000 benignos da base de dados Alexa 500 (SESHAGIRI, *et al.*, 2016). O trabalho utiliza o Google Safe Browsing para validar a rotulagem das amostras benignas e malignas. É utilizado uma abordagem estática neste trabalho. Na análise estática é utilizado um algoritmo de árvore de decisão visando estimar a probabilidade de cada nó e classificá-los entre benigno e malware. As amostras são encaminhadas para uma árvore de decisão que irá verificar se nelas exista algum conteúdo classificado como maligna (e.g. lista negra). Caso possuam, a árvore de decisão irá calcular as probabilidades e caso esteja com um valor abaixo de 20% (definido de forma arbitrária), as amostras são rotuladas como verdadeiramente maliciosas. No estágio de classificação, não são descritas quaisquer informações de configuração dos classificadores e variações de parâmetros, portanto, supõe-se que os parâmetros empregados estejam relacionados às configurações padrões da ferramenta Weka. As bases de dados não são disponibilizadas o que também dificulta a reprodutibilidade dos testes. Neste artigo, não é demonstrada a sensibilidade do teste através de uma matriz de confusão ou curva ROC ao qual dificulta verificar o real desempenho do algoritmo na tarefa de classificação. O trabalho de SESHAGIRI, *et al* (2016) atinge uma acurácia de 89,44% (SESHAGIRI, *et al.*, 2016).

JAYASINGHE, *et al* (2014) emprega 10.620 javascripts malignos obtidos de diferentes sites e 10.620 benignos da base de dados Alexa 500 (JAYASINGHE, *et al.*, 2014). O trabalho utiliza o Google Safe Browsing para validar a rotulagem das amostras benignas e malignas. O trabalho utiliza uma abordagem dinâmica. Na análise dinâmica são utilizadas árvores de decisão, Classificador de Bayes e SVM (Máquinas de Vetor de Suporte) para classificar as amostras entre benignas e malware. Primeiramente, são retirados das páginas os códigos gerados durante a abertura da página. Esses códigos são convertidos para um formato inteligível para as máquinas de aprendizagem através de redução de dados, extração de características (feitas dinamicamente através da ferramenta ADSandbox) e representação de características. Em seguida, um classificador é treinado com as amostras obtidas e por fim novas páginas são apresentadas para o classificador a fim de verificar a sua assertividade. No estágio de classificação, a validação cruzada é implementada através do método $k$-fold, $k = 10$. A base de dados é dividida randomicamente em 10 partes iguais, contendo a mesma quantidade de amostras malignas e benignas. O critério utilizado para dividir a base em 10 partes não é descrito, o que nos leva a concluir que foi definido de forma arbitrária. São utilizados 3 diferentes classificadores (árvores de decisão, SVM e classificador de bayes). Neste artigo, a sensibilidade do teste é aferida através de uma matriz de confusão e gráficos boxplots. Logo, é possível verificar o real desempenho do algoritmo na tarefa de classificação. O trabalho de JAYASINGHE, *et al* (2014) atinge uma acurácia máxima de 96,55% na tarefa de classificação com a SVM (JAYASINGHE, *et al.*, 2014).

KAPLAN, *et al* (2013) emprega 563 Javascripts malignos e 3.954 benignos obtidos de diferentes fontes (KAPLAN, *et al.*, 2013). O trabalho não descreve nenhum método para validar o rótulo das amostras. É utilizado uma abordagem estática neste trabalho. Na análise estática é utilizado um classificador Bayesiano para classificar as amostras em benignas e malignas. As amostras são encaminhadas para o classificador que irá verificar se nelas existem alguma *string* classificado como maligna (e.g. lista negra). Caso possuam, o classificador irá rotular como maligna. No estágio de classificação, não são descritas quaisquer informações de configuração dos classificadores e variações de parâmetros, portanto, supõe-se que os parâmetros empregados estejam relacionados às configurações padrões da ferramenta Weka. Os resultados não são comparados com outras abordagens, o que dificulta medir a real efetividade da abordagem propostas. As bases de dados não são disponibilizadas o que também dificulta a reprodutibilidade dos testes. Neste artigo, não é demonstrada a sensibilidade do teste através de uma matriz de confusão ou curva ROC ao qual dificulta verificar o real desempenho do algoritmo na tarefa de classificação. O trabalho de KAPLAN, et al (2013) atinge uma acurácia de 99,00% na detecção de JavaScripts obfuscados (KAPLAN, *et al.*, 2013).

Quanto às bases de dados de arquivos malware, os trabalhos do estado da arte apenas informam as fontes de aquisição dos arquivos, no entanto, não há a descrição de quais arquivos são empregados nos experimentos. Logo, torna-se inviável a réplica das obras de PEKTAS, *et al* (2017), SESHAGIRI, *et al* (2016), JAYASINGHE, *et al* (2014) e KAPLAN, *et al* (2013). O trabalho proposto cria a base de dados PAEMAL cujo objetivo é dar total possibilidade da metodologia proposta ser replicada, por terceiros, em trabalhos futuros (PAEMAL, 2019). Logo, o artigo proposto, ao disponibilizar, livremente, a sua base de dados viabiliza transparência e imparcialidade à pesquisa, além de demonstrar a veracidade dos resultados alcançados.

Na etapa de extração de características, alguns trabalhos do estado-da-arte necessitam da presença local do arquivo visando a análise do seu código fonte. Enfatiza-se que *cyber*-ataque pode ser executado em um web servidor remoto ao invés do computador pessoal. Conclui-se que a extração de características estáticas é inválida mediante aplicativos malware executados no servidor visto que não há como periciar códigos fontes remotos sem a permissão do administrador do servidor web requisitado. Então, ao invés da inexequível análise estática, o nosso antivírus realiza a análise de comportamentos (dinâmica) do computador pessoal a partir do momento em que a página web foi requisitada.

Ao todo, nossa extração dinâmica de características monitora 11,777 comportamentos suspeitos no computador pessoal provocado pelo script PHP executado no servidor remoto. Inclui-se a perícia quanto à corrupção do Sistema Operacional e do navegador, além de perícia no tráfego de rede. Nossa solução NGAV é capaz de reconstruir uma cadeia de eventos, destrinchando a real intenção do *cyber*-ataque. Logo, o nosso antivírus não se atém a eventos individuais e discretos.

Na etapa de classificação entre arquivos benignos e malware, uma boa capacidade de generalização de técnicas de máquinas de aprendizado estatístico pode depender de uma boa escolha dos seus parâmetros de configuração. Em máquinas de aprendizado, não há um conjunto de parâmetros que satisfaça todos os tipos de aplicações (HUANG, *et al.*, 2012). A melhor combinação depende do conjunto de dados empregados (HUANG, *et al.*, 2012). Um método para identificar bons parâmetros consiste em treinar diferentes combinações das máquinas de aprendizado (HUANG, *et al.*, 2012). Então, o trabalho proposto explora diferentes parâmetros das máquinas de aprendizado. A hipótese é verificar se os classificadores sofrem alterações, em suas acurácias, em função das condições iniciais. Por outro lado, as obras de PEKTAS, *et al* (2017), SESHAGIRI, *et al* (2016), JAYASINGHE, *et al* (2014) e KAPLAN, *et al* (2013) não investigam diferentes parâmetros das máquinas de aprendizado, os autores apenas empregam as configurações padrões disponibilizadas pela ferramenta Weka empregada na etapa de classificação. Conclui-se que não há garantias das máquinas de aprendizado, empregadas pelo estado-da-arte, apresentarem resultados aceitáveis caso as configurações iniciais sejam desfavoráveis.

## 4. Materiais e Métodos

O presente trabalho visa elaborar a PAEMAL (***P**HP **A**nalysis **E**nvironment Applied to **Ma**lware Machine **L**earning* - Ambiente de Análise PHP Aplicado à Aprendizagem de Máquinas de Malware). A PAEMAL é uma base de dados a qual permite a classificação de arquivos PHP entre maliciosos e benignos. A PAEMAL é composta de 200 arquivos PHP malware e outros 1000 arquivos PHP benignos. Em relação às pragas virtuais, a PAEMAL extraiu arquivos PHP maliciosos do VirusShare o qual é um repositório de amostras de malware que proporciona acesso ao código malicioso real (VIRUSSHARE, 2019). Visando o catálogo dos 200 exemplares de PHP malware, foram necessárias a aquisição e análise, por scripts autorais, de cerca de 1.300.000 (1 milhão e trezentos mil) aplicativos malware a partir dos relatórios atualizados pelo VirusShare diariamente.

No que tange aos arquivos PHP benignos, o catálogo foi dado a partir dos scripts nativos de ferramentas *open source* a exemplo do *phpMyAdmin*. Enfatiza-se que todos os arquivos benignos foram submetidos à auditoria do VirusTotal. Logo, os 1000

exemplares de arquivos PHP benignos tiveram sua benevolência atestada pelos principais antivírus comerciais mundiais. Os resultados obtidos correspondentes às análises dos PHP benignos e malware, resultante da auditoria do VirusTotal, estão disponibilizados para consulta no endereço virtual do nosso repositório (PAEMAL, 2019).

Caso não houvesse qualquer tipo de tratamento na PAEMAL, haveria uma tendência de acertos maiores na classe majoritária (benigna) e elevada taxa de erro na classe minoritária (malware). A explicação é que, na base de dados PAEMAL, a quantidade de amostras benignas e malware são desiguais; 200 e 1000, respectivamente. Logo, ao empregar bases de dados desbalanceadas, as taxas de acerto dos classificadores podem ser favorecidas basta que eles sejam tendenciosos em relação à classe majoritária (AMOR, *et al.*, 2004 ). Visando não favorecer classificadores tendenciosos, o trabalho proposto emprega uma estratégia inspirada em trabalhos de engenharia biomédica. Na área de saúde, a presença de uma anormalidade (e.g. câncer) ocorre a cada milhares de diagnósticos de pacientes sadios. Então, a estratégia biomédica diz respeito a repetir o treinamento de acordo com a razão entre a classe majoritária e minoritária (200:1000 = 5 iterações) (WANG, *et al.*, 2017). No nosso trabalho, a cada iteração, um novo pacote da classe majoritária é apresentado à classe minoritária (200:200). Dessa forma, garante-se o não favorecimento a classificadores tendenciosos aliado à manutenção da diversidade das distintas amostras, da classe majoritária, contidas na base de dados (WANG, *et al.*, 2017).

Na prática clínica biomédica, a absorção de uma amostra maligna (e.g.: câncer) acarreta em um falso negativo. Vale salientar que as chances de recuperação da paciente estão associadas à detecção da doença de maneira precoce. Então, o trabalho proposto se inspira nos cuidados metodológicos tomados pelo estado-da-arte da engenharia biomédica no sentido de reservar quantidades relevantes de exemplares benignos e malware nas amostras separadas para o treinamento e teste. Logo, supondo uma amostra reservada à teste com pouca ou nenhuma instância da classe malware, logo a classificação, tendenciosa à classe benigna, teria sua taxa de acerto favorecida. Portanto, o trabalho proposto apresenta o cuidado metodológico de selecionar equitativamente, de forma randômica, exemplares benignos e malware para as amostras destinadas ao treinamento e teste.

O objetivo da criação da base de dados PAEMAL é dar total possibilidade da metodologia proposta ser replicada, por terceiros, em trabalhos futuros. Logo, o PAEMAL disponibiliza, livremente, de todas as suas amostras tanto benignas quanto malware:

- auditorias do VirusTotal,
- análises dinâmicas da nossa *Next Generation Sandbox*,

A PAEMAL também disponibiliza, em seu endereço virtual, seus 1000 arquivos PHP benignos. Além disso, a nossa base exibe a relação de todos os outros 200 arquivos PHP, dessa vez, malware. Então, há a possibilidade da aquisição de todos os aplicativos malware, empregados pela PAEMAL, através do estabelecimento de acordo e submissão às normas de uso do *VirusShare* (VIRUSSHARE, 2019). Conclui-se que a nossa base de dados PAEMAL viabiliza transparência e imparcialidade à pesquisa, além de demonstrar a veracidade dos resultados alcançados. Então, espera-se que o PAEMAL sirva de base para a criação de novos trabalhos científicos visando novos *Next Generation Antivirus*.

## 5. Metodologia

A Figura 3 exibe o diagrama da metodologia proposta em diagrama de blocos. Inicialmente, é criada uma aplicação web empregando um script PHP suspeito no servidor. Então, o cliente requisita a página Web suspeita do servidor. A partir daí, os comportamentos maliciosos, oriundos do ataque "sem arquivos", são auditados por nossa *Web-Server Next Generation Sandbox*. Na etapa seguinte, as características dinâmicas dos arquivos PHP são armazenadas num formato compatível com o aprendizado de máquina. Como método de extração de características, alguns comportamentos, auditados pela *Sandbox* são desprezados. O critério adotado de mineração diz respeito à eliminação de características as quais dizem respeito a um único arquivo PHP, como por exemplo, nomes de processo, hashes md5 e sha, dentre outros.

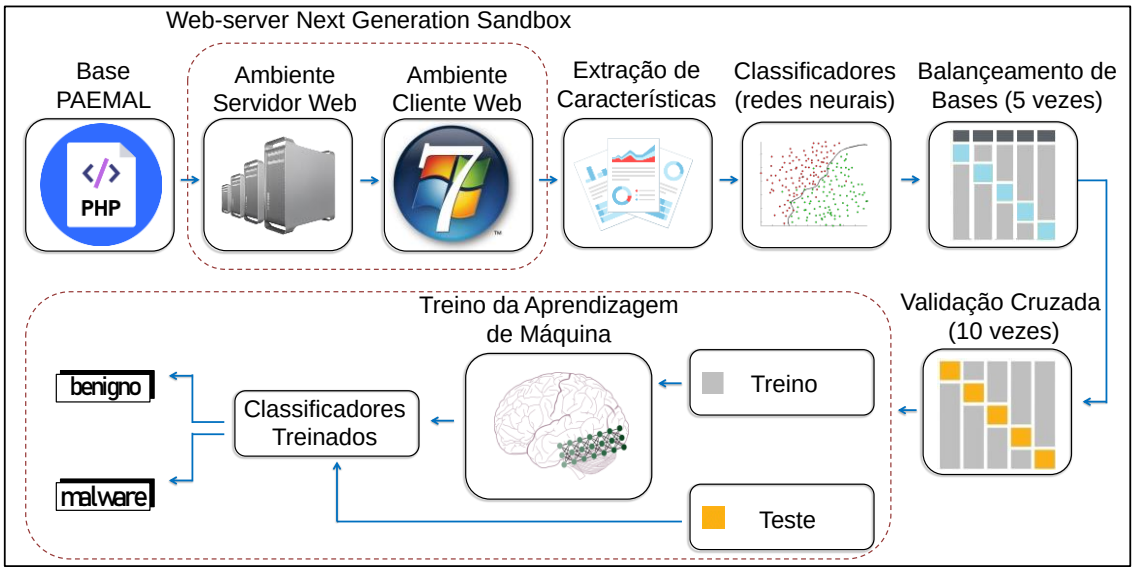



**Figura 3. Diagrama da Metodologia Proposta.**

A PAEMAL apresenta 200 e 1000 arquivos PHP malignos e benignos, respectivamente. Então, por cinco iterações, um pacote distinto de 200 exemplares da classe majoritária (benigna) é apresentado aos 200 exemplares da classe minoritária (malware). Após o balanceamento da base de dados, os comportamentos suspeitos dos arquivos PHP servem como atributos de entrada das redes neurais artificiais empregadas como classificadores. O objetivo é agrupar os arquivos PHP em duas classes; benignos e malware. Em cada combinação (200 benignos: 200 malware) oriundas do balanceamento da base de dados, é empregado o método de validação cruzada *k-fold*, onde k=10. A acurácia do classificador é a média aritmética da taxa de acertos obtida nas dez iterações.

De modo a validar o nosso NGAV, o trabalho proposto desenvolve um ambiente controlado nomeado *Web-Server Next Generation Sandbox*. Em nosso ambiente, são desenvolvidos o lado servidor e o cliente visando virtualizar o servidor web malicioso e o computador pessoal, respectivamente. O nosso servidor virtualizado é composto pelo Sistema operacional Linux, Servidor HTTP Apache, Servidor MySQL e interpretador PHP. No cliente, é empregado o Sistema Operacional Windows 7 dotado das instalações do Python, Máquina Virtual Java (JVM), Adobe Reader, Microsoft Office, e navegador Chrome. Em nossa *Web-Server Next Generation Sandbox*, os arquivos PHP, da nossa base de dados, são executados no servidor de modo a infectar, propositalmente, o cliente.

Então, os comportamentos maliciosos, oriundos do ataque “sem arquivos”, servem como atributos de entrada das máquinas de aprendizado estatístico.

## 5.1 Extração Dinâmica de Características

Em nossa metodologia, o malware é executado visando infectar, propositalmente, o Windows 7 auditado, em tempo real (dinâmico). Na nossa *Web-Server Next Generation Sandbox,* a quantidade de características está em função do comportamento dinâmico do sistema. Em média, são geradas 11.777 características referentes ao monitoramento do arquivo suspeito no ambiente controlado proposto. A seguir, são detalhados os grupos de características referentes ao monitoramento do sistema.

- ✓ Características relacionadas à Injeção de código, técnica usada por um invasor para introduzir código em programas vulneráveis e mudar seu comportamento. A auditória verifica se o servidor requisitado tenta:
  - executar um processo e injetar código enquanto é descompactado;
  - injetar código em um processo remoto com o uso de uma das seguintes funções: *CreateRemoteThread* ou *NtQueueApcThread*.
- ✓ Características relacionadas a Keyloggers, programas que gravam todas as entradas de teclado feitas pelo usuário, com a finalidade principal de capturar de forma ilegal senhas e outras informações confidenciais. Verifica se o servidor investigado tenta:
  - criar mutexes dos keyloggers *Ardamax* ou *Jintor*.
- ✓ Características relacionadas à busca de outros programas possivelmente instalados. O objetivo é verificar se o servidor auditado busca:
  - descobrir onde o browser está instalado, caso exista algum no sistema;
  - descobrir se existe algum *sniffer* ou algum analisador de pacotes de rede instalado.
- ✓ Características relacionadas a desabilitar os componentes do Windows:
  - Verifica se o servidor testado tenta desabilitar algum dos programas do Windows: terminal de comandos, gerenciador de dispositivos ou Registro (Regedit).
- ✓ Características relacionadas à forense de memória, processo em que o conteúdo da memória RAM é periciado para fins de diagnóstico. A forense digital proposta audita se o servidor requisitado tenta:
  - encontrar URLs maliciosas no processamento da forense da memória;
  - encontrar evidências da presença e uso do programa Yara. Tal programa é, em regra geral, empregado para realizar a forense digital de memória.
- ✓ Características relacionadas a mineração de cripto-moedas:
  - O nosso antivírus audita se o servidor invocado tenta se conectar a *pools* de mineração, com a finalidade de gerar moedas virtuais sem o conhecimento (e sem beneficiar) o proprietário do computador.
- ✓ Características relacionadas a modificações no sistema:

- O antivírus proposto verifica se o servidor demandado tenta criar ou modificar certificados de sistema, avisos do centro de segurança, comportamentos de controle de contas de usuário, papel de parede da área de trabalho ou valores do *ZoneTransfer.ZoneID* no identificador de zona ADS (*Alternate Data Stream*).

✓ Características relacionadas ao *Microsoft Office*. O nosso antivírus verifica se o servidor requisitado tenta:

- criar um objeto atrelado à linguagem de programação *Visual Basic*;
- executar processos do *Microsoft Office* inseridos em um objeto de interface de linha de comando empacotado.

✓ Características relacionadas a empacotamento e obfuscação. A forense digital proposta verifica se o servidor demandado:

- possui informação compactada ou criptografada indicativa de empacotamento;
- cria uma cópia ligeiramente modificada dele mesmo (pacote polimórfico);
- é compactado utilizando UPX (*Ultimate Packer for Executables*) ou VMProtect (software utilizado para obfuscar código e virtualizar programas).

- Características relacionadas à persistência. Cabe ressaltar que a vítima pode não estar livre da infecção de um malware mesmo após a sua detecção e eliminação. A persistência das malfeitorias pode ocorrer mesmo após a exclusão do malware (LIMA, *et al.*, 2018). Logo, quando o sistema operacional é inicializado, o *cyber*-ataque recomeça devido ao recomeço da vulnerabilidade explorada pelo malware (e.g.: redirecionar a página inicial do Internet Explorer). Logo, o antivírus criado audita se o servidor requisitado tenta:

  - usar Javascript em um valor de chave de registro no Registro do Sistema (Regedit);
  - instalar um auto-executável na iniciação do Windows (*autorun*);
  - instalar um executável nativo para ser executado na inicialização do Windows.

✓ Características relacionadas a POS (*Point of Sale*), tipo de ataque que visa obter as informações de cartões de crédito e de débito das vítimas. O antivírus criado audita se o servidor invocado tenta:

- criar arquivos relacionados ao malware POS Alina;
- contactar servidores relacionados ao malware POS Alina;
- contactar botnets relacionadas ao malware POS blackpos;
- criar mutexes relacionados ao malware POS decebel;
- criar mutexes e chaves de registro relacionados ao malware POS Dexter;
- criar mutexes e chaves de registro relacionados ao malware POS jackpos;
- contactar botnets relacionadas ao malware POS jackpos;
- contactar servidores relacionados ao malware POS poscardstealer;

✓ Características relacionadas à injetores de códigos de *powershell*. O nosso antivírus verifica se o servidor requisitado:

- é um script de powershell malware do tipo powerfun ou powerworm.
- tenta criar um processo powershell suspeito;
- tenta criar entradas de registros via scripts powershell;

- ✓ Características relacionadas aos processos. O antivírus proposto verifica se o servidor invocado:
  - interessa-se em algum processo específico em execução;
  - procura repetidas vezes por um processo não encontrado;
  - tenta falhar algum processo em específico.
- ✓ Características relacionadas a *ransomwares*, ataques que tornam os dados do equipamento inacessíveis, exigindo pagamento para restabelecer o acesso do usuário. O nosso antivírus verifica se o servidor requisitado tenta:
  - criar mutexes do *ransomware* chanitor;
  - executar comandos no bcdedit (ferramenta de linha de comando que gerencia dados de configuração de inicialização) relacionados à ransomware;
  - adicionar extensões de arquivos reconhecidamente relacionadas à *ransomwares* a arquivos que foram criptografados;
  - executar movimentações em arquivos, que podem ser indicativos do processo de encriptação de dados visto em um ataque *ransomware*;
  - criar instruções de como reverter a criptografia feita em um ataque *ransomware* ou se tenta gerar um arquivo de chave;
  - escrever uma mensagem de resgate em disco, provavelmente associada a um ataque *ransomware*;
  - esvaziar a lixeira;
  - remover ou desabilitar o *shadow copy*, recurso que tem como finalidade agilizar a restauração de dados, a fim de evitar a recuperação do sistema.
- ✓ Características relacionadas ao uso de sandboxes. A forense digital averigua se o servidor invocado tenta:
  - detectar se as sandboxes: Cuckoo, Joe, Anubis, Sunbelt, ThreatTrack/GFI/CW ou Fortinet estão sendo utilizadas, por meio da presença de arquivos próprios utilizados por elas;
  - procurar por diretórios conhecidos onde uma sandbox pode executar amostras;
  - checar se alguma atividade humana está sendo desempenhada;
  - descobrir o tempo em espera do Windows, a fim de determinar o tempo total de atividade do Windows;
  - instalar um procedimento que monitora eventos do mouse;
  - desligar ou reiniciar o sistema visando burlar a *sandbox*;
  - atrasar as tarefas de análise;
  - desligar funções do Windows monitoradas pela *Cuckoo sandbox*.
- ✓ Características relacionadas ao Registro (Regedit) do Windows 7 SO:
  - Mudanças nas associações entre extensões de arquivos e conjunto de *software* instalado na máquina (HKEY_CLASSES_ROOT).
  - Modificações nas informações sobre o usuário atual (HKEY_CURRENT_USER).
  - Corrupção do funcionamento dos drivers (HKEY_LOCAL_MACHINE).
  - Alterações nas configurações de aparência do Windows e as configurações efetuadas pelos usuários, como papel de parede, protetor de tela e temas (HKEY_USERS).

- Mudanças nas Configurações de hardware (HKEY_CURRENT_CONFIG).

✓ Características relacionadas a cavalos de troia (programa malicioso que entra em um computador disfarçado como outro programa, legítimo) de acesso remoto, ou RAT (Remote Access Trojans). O nosso antivírus verifica se o servidor requisitado tenta criar arquivos, chaves de registro e/ou mutexes relacionados aos RATs: Adzok, bandook, beastdoor, beebus, bifrose, blackhole/schwarzesonne, blackice, blackshades, bladabindi, bottilda, bozokrat, buzus, comrat, cybergate, darkcloud, darkshell, delf trojan, dibik/shark, evilbot, farfli, fexel, flystudio, fynloski/darkcomet, ghostbot, hesperbot, hkit backdoor, hupigon, icepoint, jewdo backdoor, jorik trojan, karakum/saharabot, koutodoor, aspxor/kuluoz, likseput, madness, madness, magania, minerbot, mybot, naid backdoor, nakbot, netobserve spyware, netshadow, netwire, nitol/servstart, njrat, pasta trojan, pcclient, plugx, poebot/zorenium, poison ivy, pincav/qakbot, rbot, renos trojan, sadbot, senna spy, shadowbot, siggen, spynet, spyrecorder, staser, swrort, travnet, tr0gbot bifrose, turkojan, urlspy, urx botnet, vertexnet, wakbot, xtreme, zegost.

✓ Características relacionadas ao *payload* na rede. O nosso antivírus audita se o servidor invocado tenta:

- verificar se a atividade de rede contém mais de um *useragent* único;
- criar mutexes de protocolo de conexão de área de trabalho remota (RDP);
- checar a presença de clientes de chat mIRC;
- instalar Tor (the onion router, software de código aberto com a capacidade de criar de forma segura e anonimamente conexões online, a fim de resguardar o direito à privacidade do usuário), ou um serviço oculto Tor na máquina;
- conectar a um encurtador de URL chinês com histórico malicioso;
- criar mutexes relacionados a ferramentas de administração remota VNC (Virtual Remote Computer).

✓ Características relacionadas ao tráfego de rede. Audita-se se o servidor suspeito tenta:

- conectar-se a um IP que não está mais respondendo a requisições;
- resolver um domínio de topo suspeito;
- iniciar a escuta (socket) com algum servidor;
- conectar a algum domínio de DNS dinâmico;
- fazer requisições de HTTP;
- gerar tráfego ICMP;
- conectar-se a algum servidor de IRC (possivelmente parte de alguma botnet);
- fazer requisições SMTP (possivelmente envio de SPAM);
- conectar-se a algum serviço oculto TOR por meio de um gateway TOR;
- iniciar o arquivo wscript.exe, que pode indicar um script baseado em download de *payload* (corpo do pacote);
- gerar alertas IDS ou IPS, com Snort e Suricata (ferramentas de gerenciamento e monitoramento de redes).

✓ Características relacionadas à servidores DNS (Domain Name System, servidores responsáveis pela tradução de endereços URL em IP). O nosso antivírus investiga se o servidor requisitado tenta:

- conectar a servidores DNS de provedores de DNS dinâmicos;
- conectar ao site malicioso expirado 3322.org, ou ao seu domínio relacionado, 125.77.199.30;
- resolver algum domínio *Free Hosting*, possivelmente malicioso.

## 5.2 Redes Neurais visando o Reconhecimento de Padrão de Malware

Redes neurais são modelos, de inteligência computacional, utilizadas para resolver problemas de classificação tendo como principal característica o poder de generalização diante de dados não apresentados à rede. A rede ELM (*Extreme Learning Machine* – Máquina de Aprendizado Extremo) tem como principal característica a velocidade de treinamento e predição de dados comparada a outros classificadores (HUANG, *et al.*, 2012). As ELMs têm sido largamente aplicadas nas mais diversas áreas como na Engenharia Biomédica (AZEVEDO, *et al.*, 2015) (AZEVEDO, *et al.*, 2015b) (LIMA, *et al.*, 2016) (LIMA, *et al.*, 2014 ) (CORDEIRO, *et al.*, 2012).

As redes ELMs podem contribuir bastante para o avanço da segurança em dispositivos visto que a inteligência artificial ainda se encontra em um estágio inicial na área de Segurança da Informação (HENKE, *et al*., 2011). A rede ELM tem como principal característica a velocidade de treinamento e predição de dados comparada a outros classificadores (HUANG, ZHOU, *et al.*, 2012). A rede ELM é uma rede de camada escondida única, não recursiva. O processo de aprendizagem da rede ELM é baseado na inversa generalizada de Moore-Penrose (pseudo-inversa), onde são calculados os pesos entre a camada escondida e a camada de saída (HUANG, ZHOU, *et al.*, 2012).

A aprendizagem da rede ELM é realizada em lote, onde todos os dados são apresentados à rede antes do ajuste dos pesos referentes às ligações sinápticas entre os neurônios da camada escondida e de saída. Há uma única iteração, tornando o treinamento mais rápido do que as abordagens convencionais. Então, não é necessário determinar o máximo número de iterações, uma vez que o algoritmo não é iterativo. Além disso, por não se basear no método de gradiente descendente, a rede não sofre o problema de mínimo local nem é necessária a definição de um parâmetro de taxa de aprendizagem.

Matematicamente, na rede ELM os atributos de entrada $x_{ik}$ correspondem ao conjunto $\{x_{it} \in R;\ i \in N^*,\ i = 1, \dots, n;\ t \in N^*,\ t = 1, .. v\}$. Logo, há *n* características extraídas da aplicação e $v$ vetores de dados de treinamento. A camada escondida $h_j$, constituída por $m$ neurônios, é representada pelo conjunto $\{h_j \in R;\ j \in N^*,\ j = 1, \dots, m\}$.

O processo de treinamento da ELM é rápido por ser composto por poucas etapas. Inicialmente, os pesos de entrada $w_{ji}$ e $biasb_{jt}$ são definidos de maneira aleatória. Dada uma função de ativação $f: \mathbb{R} \to \mathbb{R}$, o processo de aprendizagem é dividido em três passos:

1. Atribuição aleatória de pesos $w_{ji}$, correspondente aos pesos entre a camada de entrada e a camada escondida, e *bias* $b_{jk}$.
2. Calcular a matriz H, que corresponde à saída dos neurônios da camada escondida.

3. Calcular a matriz dos pesos de saída $\beta = H^{\dagger}Y$, onde $H^{\dagger}$ é a matriz inversa generalizada de Moore-Penrose da matriz H, e $Y$ corresponde à matriz de saídas desejadas *s*.

A saída dos neurônios da camada escondida, correspondente à matriz H, é calculada através da função de ativação, entradas e pesos da camada escondida, conforme mostra a Equação (1).

$$H_{jt} = \begin{bmatrix} K(11) & \cdots & K(1N) \\ \vdots & \ddots & \vdots \\ K(V1) & \cdots & K(VN) \end{bmatrix} \quad (1)$$

Diferente das redes com retropropagação, na rede ELM não é necessário definir critério de parada para treinamento nem criar mecanismos para que a rede não perca a capacidade de generalização. O motivo é a rede ELM apresenta uma única iteração. Desse modo, não é necessária a separação de conjunto de dados em treinamento, validação e teste. Basta a divisão em conjuntos de treinamento e teste, permitindo um maior número de amostras para esses dois conjuntos em comparação a redes neurais baseadas em retropropagação. Uma vez treinada a rede, os padrões de teste são apresentados juntamente com a saída desejada. A rede não sofrerá mais ajustes e apenas calculará o resultado obtido para cada conjunto de teste apresentado. Ao comparar os dados esperados com os obtidos é avaliado o grau de precisão da rede ELM.

O trabalho explora 9 (nove) tipos distintos de *kernels* visando redes neurais ELMs. No estado-da-arte, HUANG, *et al*, (2012) descreve 7 (sete) desses *kernels*; Linear, Polinomial, Transformada *Wavelets*, Sigmoid, Senoidal, *Hard* Limite e Tribas (*Triangular Base Function*). Além disso, são empregados dois outros *kernels* validados no campo da Engenharia Biomédica: *Fuzzy*-Dilatação e *Fuzzy*-Erosão (AZEVEDO, *et al.*, 2015) (AZEVEDO, *et al.*, 2015b).

Os *kernels* Polinomial, *Wavelets* e Linear não empregam camadas escondidas (HUANG, *et al.*, 2012). Nesses *kernels*, os cálculos são baseados na transformação dos dados de entrada e podem trabalhar de maneira aproximada dos *kernels* contendo arquiteturas dotadas de camadas escondidas (HUANG, *et al.*, 2012). Nos referidos *kernels*, uma boa capacidade de generalização da rede ELM depende de uma escolha ajustada dos parâmetros $(C, \gamma)$. Então, há a investigação dos parâmetros $(C, \gamma)$ inspirada no método, proposto por HUANG, *et al*, (2012), que consiste em treinar sequencias crescentes de $C$ e $\gamma$. Matematicamente, $2^n$, onde $n = \{-24, 10, 0, 10, 25\}$. A hipótese é verificar se esses parâmetros com valores diferentes aos padrões; $(C = 1,\ \gamma = 1)$, geram resultados superiores. No *kernel* Linear, há a investigação apenas do parâmetro de custo $C$ visto que não cabe a exploração do parâmetro do kernel $\gamma$ (HUANG, *et al.*, 2012).

Os *kernels* Sigmoidal, Senoidal, *Hard* Limite, *Tribas*, *Fuzzy*-Dilatação e *Fuzzy*-Erosão empregam arquiteturas dotadas de camadas escondidas. Então, há a investigação quanto à quantidade de neurônios na camada escondida desses *kernels*. A hipótese é verificar se arquiteturas que exijam um maior volume de cálculos, como por exemplo, dobrar a quantidade de neurônios na camada escondida, são capazes de gerar taxas de acertos superiores em comparação a arquiteturas que exijam uma menor quantidade de cálculos. Há a avaliação de 2 (duas) arquiteturas, elas empregam 100 e 500 neurônios em suas respectivas camadas escondidas. Nós investigamos 30 conjuntos diferentes de pesos

iniciais referentes às ligações sinápticas entre os neurônios. A semente do gerador aleatório varia de 1 a 30 incrementalmente.

## 6. Resultados

A Tabela 3 detalha os resultados obtidos pelas redes neurais ELM através dos *kernels Wavelets* e Polinomial. Em cada *kernel*, por 5 vezes, um pacote distinto de exemplares benignos (classe majoritária) é apresentado ao pacote de exemplares malware (classe minoritária). Em cada uma dessas 5 vezes, há a validação cruzada através do método *k-fold* onde $k = 10$. Então, há 50 (5*10) iterações em cada linha da Tabela 3. Em relação à precisão na fase de teste, o melhor desempenho médio foi de 97,50% através do *kernel* Polinomial dotado dos parâmetros $(C, \gamma) = (2^{-10}, 2^{0})$. Nessa nossa melhor configuração, o tempo de treinamento médio é de apenas 0,04 segundos acompanhado de um desvio padrão de 0,05. Na Tabela 3, na Tabela 4 e na Tabela 5, há apenas a descrição dos melhores e piores casos, nessa ordem, para cada *kernel* ELM.

**Tabela 3. Resultado das redes ELMs. Os parâmetros $(C, \gamma)$ variam de acordo com o conjunto $\{2^{-24}, 2^{-10}, 2^{0}, 2^{10}, 2^{25}\}$.**

| *Kernel* | $(C, \gamma)$ | *Acerto no treino (%)* | *Acerto no teste (%)* | *Tempo de treino (seg.)* | *Tempo de teste (seg.)* |
|---|---|---|---|---|---|
| Polynomial | $(2^{-10}, 2^{0})$ | 97,40 ± 2,30 | **97,50 ± 4,32** | **0,04 ± 0,05** | **0,03 ± 0,03** |
| | $(2^{-24}, 2^{10})$ | 49,95 ± 0,11 | 49,95 ± 0,34 | 0,09 ± 0,10 | 0,04 ± 0,05 |
| Wavelets | $(2^{10}, 2^{0})$ | **100,00 ± 0,00** | 66,59 ± 12,39 | 0,07 ± 0,08 | 0,05 ± 0,06 |
| | $(2^{-24}, 2^{-24})$ | **100,00 ± 0,00** | 52,35 ± 4,35 | 0,07 ± 0,09 | 0,05 ± 0,06 |

**Tabela 4. Resultado das redes ELMs dotadas do *kernel* Linear. O parâmetro $C$ varia de acordo com o conjunto $\{2^{-24}, 2^{-10}, 2^{0}, 2^{10}, 2^{25}\}$.**

| *Kernel* | $C$ | *Acerto no treino (%)* | *Acerto no teste (%)* | *Tempo de treino (seg.)* | *Tempo de teste (seg.)* |
|---|---|---|---|---|---|
| Linear | $2^{-24}$ | **98.22 ± 1.24** | **97.30 ± 5.51** | **0.04 ± 0.05** | 0.03 ± 0.04 |
| | $2^{10}$ | 99.95 ± 0.11 | 87.30 ± 14.04 | **0.04 ± 0.05** | **0.03 ± 0.03** |

A Tabela 4 exibe os resultados alcançados pelas redes ELMs dotadas do *kernel* Linear. Há apenas a investigação do parâmetro de custo $C$, não é possível a exploração do parâmetro $\gamma$ em um *kernel* Linear (HUANG, *et al.*, 2012). Cada linha na Tabela 4 contém 50 iterações assim como ocorreu na Tabela 3. Em relação à precisão na fase de teste, a máxima e mínima precisão média foi de 97,30% e 87,30%, respectivamente. Conclui-se que o parâmetro de custo $C$ é capaz de aperfeiçoar o desempenho do *kernel* Linear quando aplicado à detecção de malware.

Tabela 5 detalha os resultados obtidos pelas redes neurais ELMs dotadas de camada escondida. Em cada kernel, por 5 vezes, um pacote separado de amostras benignas (classe majoritária) é apresentado ao pacote de amostras de malware (classe minoritária). Em cada uma dessas 5 vezes, investigamos 30 conjuntos diferentes de pesos iniciais referentes às ligações sinápticas entre os neurônios. A semente do gerador aleatório varia de 1 a 30 incrementalmente. Então, para cada conjunto de pesos sinápticos é empregado o método k-fold, onde $k = 10$. Então, há 1500 (5 * 30 * 10) iterações em cada linha da Tabela 5. Com relação à precisão, o desempenho médio máximo foi de 81,99% através do kernel de Fuzzy-Erosão dotado de 500 neurônios em sua camada escondida.

A Figura 4 e Figura 5 são representações gráficas dos resultados descritos na Tabela 3, na Tabela 4 e na Tabela 5. A Figura 4 (a) e a Figura 4 (b) mostram os *boxplots* para uma precisão ótima durante a fase de treinamento e teste, respectivamente. As taxas de acertos ótimas e suas configurações estão presentes nas primeiras linhas de cada kernel presentes na Tabela 3, na Tabela 4 e na Tabea 5.

**Tabela 5. Resultados das redes ELMs. O número de neurônios da camada escondida varia de acordo com o conjunto {100, 500}.**

| *Kernel* | Neurons | *Acerto no treino (%)* | *Acerto no teste (%)* | *Tempo de treino (seg.)* | *Tempo de teste (seg.)* |
|---|---|---|---|---|---|
| Sigmoid | 500 | 77.37 ± 22.25 | 61.05 ± 14.08 | 0.11 ± 0.11 | 0.01 ± 0.02 |
| | 100 | 77.18 ± 22.10 | 60.71 ± 14.83 | 0.03 ± 0.03 | **0.00 ± 0.01** |
| Seno | 500 | **100.00 ± 0.00** | 59.12 ± 14.41 | 0.12 ± 0.12 | 0.01 ± 0.02 |
| | 100 | 89.44 ± 13.02 | 58.48 ± 14.20 | 0.03 ± 0.03 | **0.00 ± 0.01** |
| *Hard* limite | 100 | 50.05 ± 0.11 | 50.05 ± 0.33 | **0.02 ± 0.03** | **0.00 ± 0.01** |
| | 500 | 50.05 ± 0.11 | 50.05 ± 0.33 | 0.12 ± 0.11 | 0.01 ± 0.02 |
| *Tribas* | 100 | 50.18 ± 0.41 | 49.77 ± 1.36 | **0.02 ± 0.03** | **0.00 ± 0.01** |
| | 500 | 50.46 ± 0.55 | 49.61 ± 2.14 | 0.10 ± 0.10 | 0.01 ± 0.02 |
| *Fuzzy*-Dilatação | 100 | 100.00 ± 0.02 | 80.64 ± 18.85 | **0.02 ± 0.03** | **0.00 ± 0.01** |
| | 500 | **100.00 ± 0.00** | 77.25 ± 15.57 | 0.12 ± 0.12 | 0.01 ± 0.02 |
| *Fuzzy*-Erosão | 500 | **100.00 ± 0.00** | **81.99 ± 17.03** | 0.15 ± 0.14 | 0.04 ± 0.04 |
| | 100 | **100.00 ± 0.00** | 80.75 ± 18.76 | 0.04 ± 0.04 | 0.01 ± 0.02 |

A Figura 4 (a) exibe as acurarias resultantes do treinamento. Entre as melhores configurações de cada *kernel*, a melhor e pior precisão média foram de 100,00% e 50,05% através dos *kernels Wavelets* e *Hard* Limite, respectivamente. A Figura 4 (b) exibe as acurarias resultantes da fase de teste. Entre as melhores configurações de cada kernel, a melhor e pior precisão média foram de 97,50% e 49,77% através dos kernels Polinomial e Tribas (*Triangular Basis Function*), respectivamente. Conclui-se que, na fase de teste, a melhor abordagem é quase 100% superior ao pior cenário possível. Portanto, a escolha de um kernel adequado, composto por uma arquitetura correta, é essencial para maximizar a acurácia quanto à identificação de malware.

A Figura 5 (a) e a Figura 5 (b) mostram os boxplots dos tempos gastos durante as fases de treinamento e teste, respectivamente. Em relação ao tempo de treinamento, o nosso kernel *Fuzzy*-Erosão, contendo 500 neurônios em sua camada escondida, é mais lento em relação aos demais, quando houve o consumo médio de 0,15 segundos. Por outro lado, os *kernels Hard* limite, Tribas e *Fuzzy*-Dilatação, contendo 100 neurônios em suas camadas escondidas, apresentam o treinamento mais rápido, quando houve um consumo médio de 0,02 segundos. Em relação ao tempo consumido, durante a fase de teste, não há grande discrepâncias entre os *kernels*.

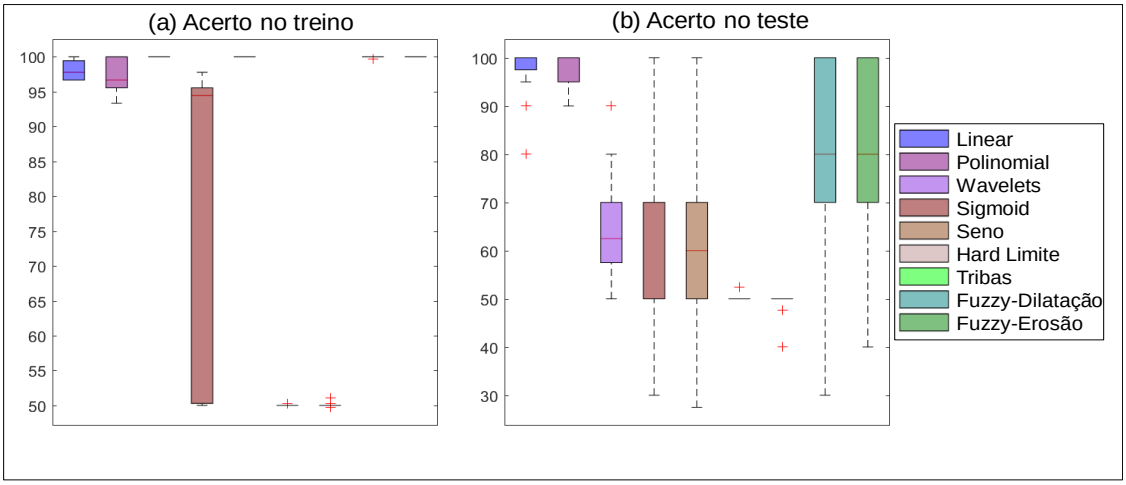



**Figura 4. (a) Boxplot referente à acurácia de treinamento. (b) Boxplot referente à acurácia de teste.**

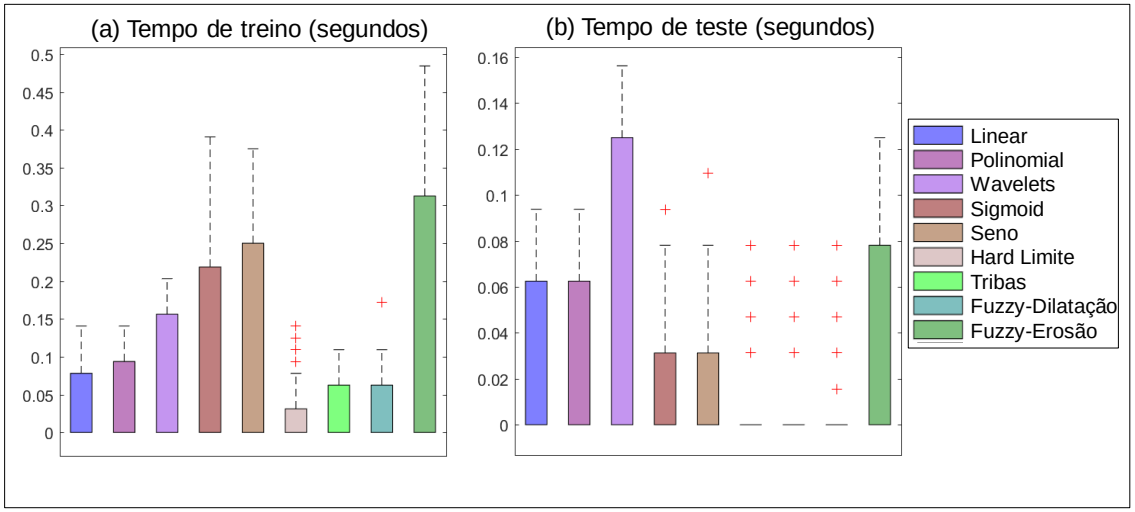



**Figura 5. (a) Boxplot referente ao tempo gasto durante o treinamento. (b) Boxplot referente ao tempo consumido durante à fase de teste.**

A

Tabela 6 exibe as matrizes de confusão das melhores configurações das redes ELMs apresentadas na Tabela 3, na Tabela 4 e na Tabela 5. A matriz de confusão assume papel importante no sentido verificar a qualidade de uma aprendizagem supervisionada. Na

Tabela 6, “B” e “M” são abreviaturas de Benigno e Malware. As classes desejadas estão dispostas no rótulo vertical enquanto as classes obtidas estão no rótulo horizontal. Na

Tabela 6, por exemplo, o *kernel* Polinomial classificou em média, de maneira equivocada, 2,28 casos como benignos quando se tratavam de malware. Ainda quanto ao *kernel* Polinomial, houve a classificação média de 0,06 casos equivocadamente ditos como malware quando se tratavam de amostras benignas. Na matriz de confusão, a diagonal principal é ocupada por casos onde a classe obtida coincide com a classe desejada. Então, um bom classificador deve ter uma diagonal principal ocupada por valores altos enquanto as demais posições devem possuir valores baixos. Na

Tabela 6, as diagonais principais estão em negrito.

**Tabela 6 Matrizes de confusão das redes neurais ELMs apresentadas na Tabela 3, na Tabela 4 e na Tabela 5**

.

| . *Kernel* | *Melhor Configuração* | | *Treino* | | *Teste* | |
|---|---|---|---|---|---|---|
| | | | B | M | B | M |
| Polinomial | $(C,\gamma) = (2^{-10}, 2^{0})$ | B | **98.94 ± 66.86** | 0.06 ± 0.24 | **11.00 ± 7.42** | 0.00 ± 0.00 |
| | | M | 2.28 ± 2.06 | **97.08 ± 69.14** | 0.28 ± 0.50 | **10.76 ± 7.64** |
| *Wavelets* | $C = (2^{10}, 2^{0})$ | B | **99.00 ± 66.81** | 0.00 ± 0.00 | **10.64 ± 7.75** | 0.36 ± 0.69 |
| | | M | 0.00 ± 0.00 | **99.36 ± 67.26** | 8.16 ± 7.58 | **2.88 ± 1.12** |
| Linear | $C = 2^{-24}$ | B | **98.88 ± 66.91** | 0.12 ± 0.48 | **10.94 ± 7.48** | 0.06 ± 0.24 |
| | | M | 2.16 ± 0.98 | **97.20 ± 67.19** | 0.30 ± 0.61 | **10.74 ± 7.50** |
| Sigmoid | $neurônios = 500$ | B | **28.18 ± 29.36** | 70.82 ± 85.51 | **2.34 ± 3.10** | 8.66 ± 8.93 |
| | | M | 3.58 ± 25.07 | **95.78 ± 66.98** | 1.27 ± 3.12 | **9.77 ± 8.06** |
| Seno | $neurônios = 500$ | B | **99.00 ± 66.16** | 0.00 ± 0.00 | **6.21 ± 3.46** | 4.79 ± 4.56 |
| | | M | 0.00 ± 0.00 | **99.36 ± 66.60** | 5.30 ± 4.19 | **5.74 ± 3.99** |
| *Hard* Limite | $neurônios = 100$ | B | **0.00 ± 0.00** | 99.00 ± 66.16 | **0.00 ± 0.00** | 11.00 ± 7.35 |
| | | M | 0.00 ± 0.00 | **99.36 ± 66.60** | 0.00 ± 0.00 | **11.04 ± 7.41** |
| Tribas | $neurônios = 100$ | B | **99.00 ± 66.16** | 0.00 ± 0.00 | **10.98 ± 7.37** | 0.02 ± 0.13 |
| | | M | 99.02 ± 66.51 | **0.34 ± 0.47** | 11.04 ± 7.41 | **0.00 ± 0.00** |
| *Fuzzy*-Dilatação | $neurônios = 100$ | B | **99.00 ± 66.16** | 0.00 ± 0.00 | **10.26 ± 7.98** | 0.74 ± 0.90 |
| | | M | 0.01 ± 0.08 | **99.35 ± 66.59** | 1.22 ± 1.22 | **9.82 ± 8.39** |
| *Fuzzy*-Erosão | $neurônios = 500$ | B | **99.00 ± 66.16** | 0.00 ± 0.00 | **10.35 ± 7.90** | 0.65 ± 0.79 |
| | | M | 0.00 ± 0.00 | **99.36 ± 66.60** | 1.19 ± 1.09 | **9.85 ± 8.31** |

A Tabela 7 disponibiliza os testes de hipóteses *t-students* (paramétrico) e Wilcoxon (não-paramétrico). Os testes se dão entre o *kern*el Polinomial e todos os demais investigados. As amostras dizem respeito à precisão durante a fase de teste com os kernels configurados em seus parâmetros ótimos apresentados nas primeiras linhas de cada *kernel*, na Tabela 3, na Tabela 4 e na Tabela 5. Observa-se que a escolha do *kernel* Polinomial foi devido ao fato dele ter alcançado a melhor precisão média, durante a fase de teste, dentre os *kernels* ELMs. O p-valor do teste é um valor escalar no intervalo [0,1]. $p$ é a probabilidade do teste estatístico apresentar o valor observado ou algo mais extremo. Se $p$ apresentar um valor baixo, o resultado observado é estatisticamente relevante. Caso o teste de hipótese seja 1, a hipótese nula é rejeitada, portanto, as distribuições são diferentes.

Ao observar a Tabela 7, os resultados do *kernel* Polinomial são confrontados contra todas as demais amostras. A hipótese foi rejeitada em quase todos os casos. Logo, em regra geral, o *kernel* Polinomial é estatisticamente distinto aos demais. A explicação é que a hipótese foi igual a 1 tanto no teste paramétrico quanto no não-paramétrico. A exceção ocorreu envolvendo os *kernels* Polinomial e Linear. Nessa exceção, a hipótese

nula foi aceita. Então, os *kernels* Polinomial e Linear geram resultados estatisticamente equivalentes tanto no teste paramétrico (*t-students*) quanto no teste não-paramétrico (*Wilcoxon*).

**Tabela 7: Teste de hipóteses *t*-students e Wilcoxon entre a melhor configuração (*kernel* Polinomial) e todas as demais.**

| ***Comparação*** | ***t-students (teste paramétrico)*** | | ***Wilcoxon (teste não-paramétrico)*** | |
|---|---|---|---|---|
| | ***Hipótese*** | ***Valor p*** | ***Hipótese*** | ***Valor p*** |
| Polinomial *vs Wavelets* | 1 | 2.08114e-22 | 1 | 6.46726e-18 |
| Polinomial *vs* Linear | 0 | 0.778313 | 0 | 0.894246 |
| Polinomial *vs Sigmoid* | 1 | 0.000000 | 1 | 0.000000 |
| Polinomial *vs* Seno | 1 | 0.000000 | 1 | 0.000000 |
| Polinomial *vs Hard* limite | 1 | 0.000000 | 1 | 0.000000 |
| Polinomial *vs Fuzy*-Dilatação | 1 | 4.91637e-194 | 1 | 6.91101e-148 |
| Polinomial *vs Fuzzy*-Erosão | 1 | 6.2273e-196 | 1 | 2.1272e-141 |

## 7. Conclusão

Apesar da presença, quase totalitária, dos antivírus nos computadores pessoais, os aplicativos malware vêm causando prejuízos bilionários e em escalas cada vez maiores (MICROSOFT, 2017). Uma das explicações é que os *cyber*-ataques se renovam sistematicamente (SOPHOS, 2014). Visando suprir as limitações dos antivírus comerciais, o estado-da-arte emprega a análise do código-fonte do arquivo suspeito, conhecida como análise estática, de modo que o repertório de instruções pode ser estudado. Portanto, é possível investigar a intenção maliciosa do arquivo antes mesmo dele ser executado pelo usuário (LIMA, *et al.*, 2018). A análise estática, no entanto, é impraticável mediante ataques "sem arquivos" visto que o arquivo é executado remotamente e, portanto, o executável não está presente no computador pessoal.

Ao invés da inexequível análise estática, a extração de características do nosso NGAV diz respeito à perícia do comportamento anômalo, no computador da vítima. Em média, nossa extração dinâmica de características monitora 11,777 comportamentos que o ataque "sem arquivos" possa fazer quando diretamente lançado de um servidor malicioso para um serviço responsivo em um computador pessoal. Ao invés de analisar eventos individuais, nossa solução consegue reconstruir a cadeia de eventos tal qual o malfeitor lançaria.

Nesse trabalho, são aplicadas máquinas de aprendizado do tipo ELM na perícia forense digital especificamente no reconhecimento de padrão de malware. Então, os comportamentos maliciosos, obtidos através da nossa *Web-server Next Generation Sandbox*, servem como atributos de entrada das máquinas de aprendizado estatístico empregadas como classificadores. A meta é agrupar os arquivos em duas classes: benigna e malware. Quanto à precisão, o *kernel* Polinomial demonstrou melhor desempenho em relação aos demais classificadores analisados, e obteve um acerto médio de 97,50%. Essa abordagem possui os seguintes parâmetros $(C, \gamma) = (2^{-10}, 2^{0})$. Por outro lado, o kernel Tribas (*Triangular Basis Function*) obteve a pior acurácia com desempenho médio de

49,77% mesmo em sua melhor configuração. Esse caso com menor precisão se dá quando o *kernel* Tribas faz uso de 100 neurônios na sua camada escondida. Conclui-se que a melhor abordagem é superior em quase 100% em comparação ao pior cenário. Logo, a escolha de uma adequada função de aprendizado, composta por corretos parâmetros, é essencial para maximizar a precisão quanto à identificação de malware.

O NGAV proposto pode ser estendido no sentido de prover *cyber*-proteção a redes locais. Logo, a meta futura é que o nosso NGAV possa ser executado tanto nos computadores pessoais quanto no servidor *proxy* o qual se constitui como o intermediário entre a rede mundial de computadores e a rede local. Caberá ao nosso futuro NGAV, executado no *proxy*, monitorar o tráfego de rede. Dessa forma, será minimizada a carga de trabalho do nosso NGAV quando executado no computador pessoal. Para tal, faz-se necessária a criação de uma nova *Web-Server Next Generation Sandbox* dotada de uma arquitetura composta por um servidor web, um servidor *proxy* e múltiplos computadores pessoais. O objetivo futuro da nossa NVAG é suprir as limitações dos mecanismos de defesa dos *proxies* os quais são baseados em listas negras assim com os antivírus comerciais. Logo, não será mais necessário aguardar que a rede local seja infectada e, em sequência haja a denuncia de comportamentos anômalos para, então, tomar-se providências quanto à detecção de um novo servidor web malicioso.

## Referências

AMOR, N. B.; BENFERHAT, S.; ELOUEDI, Z. Naive bayes vs decision trees in intrusion detection systems., In Proceedings of the 2004 ACM symposium on Applied computing, pág. 420–424., 2004.

AZEVEDO, W.; LIMA, S.; FERNANDES, I.; ROCHA, A. . F. R.; SILVA-FILHO, A..; SANTOS, W. Fuzzy Morphological Extreme Learning Machines to detect and classify masses in mammograms, IEEE International Conference on Fuzzy Systems (FUZZIEEE), Istanbul., 2015.

AZEVEDO, W.; LIMA, S.; FERNANDES, I.; ROCHA, A. . F. R.; SILVA-FILHO, A.; SANTOS, W. Morphological extreme learning machines applied to detect and classify masses in mammograms, International Joint Conference on Neural Networks (IJCNN), Killarney., 2015b.

CONRAD, E.; MISENAR, S.; FELDMAN, J., Eleventh Hour CISSP (Certified Information Systems Security Professional). Elsevier, 2017.

CORDEIRO, F. R.; LIMA, S. M. L.; SILVA-FILHO; SANTOS, W. P. Segmentation of Mammography by Applying Extreme Learning Machine in Tumor Detection, In: International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), 2012, Natal., 2012.

HUANG, G. B.; ZHOU, H.; DING, X. E.; ZHANG, R. Extreme Learning Machine for Regression and MultiClass Classification. IEEE Transactions on Systems, Man, and Cybernetics. Volume 42, número 2, pág. 513-529, 2012.

INTEL. McAfee Labs: Threat Report. Threats Statistics: Malware, Incidents, Web and Network Threats., 2018.

JAYASINGHE, G.; S., C. J.; BERTOK, P. Efficient and effective realtime prediction of drive-by download attacks. , Journal of Network and Computer Applications 38 (2014) 135–149, 2014.

KAPLAN, S.; SIEFERT, C. NOFUS: Automatically Detecting" + String.fromCharCode(32) + "ObFuSCateD".toLowerCase() + "JavaScript Code"., Microsoft Research Technical Report. MSR-TR-2011-57., 2013.

LIMA, S. M. L.; SILVA, H. K. D. L.; LUZ, J. H. D. S.; SILVA, S. L. D. P.; LIMA, H. J. D. N.; ANDRADE, A. B. A. D.; SILVA., A. M. D. Antivírus dotado de Rede Neural Artificial visandoDetectar MalwaresPreventivamente, iSys: Revista Brasileira de Sistemas de Informação (Brazilian Journal of Information Systems), 11(4), 31-62., 2018.

LIMA, S. M. L.; SILVA-FILHO, A. G.; SANTOS, W. P. A methodology for classification of lesions in mammographies using Zernike Moments, ELM and SVM Neural Networks in a multi-kernel approach , In: 2014 IEEE International Conference on Systems, Man and Cybernetics SMC, San Diego., 2014.

LIMA, S. M. L.; SILVA-FILHO, A. G.; SANTOS, W. P. Detection and classification of masses in mammographic images in a multi-kernel approach., Computer Methods and Programs in Biomedicine. 134, (2016), 11-29., 2016.

MICROSOFT. Microsoft Computing Safety Index WorldWide Report, Disponível em: https://news.microsoft.com/pt-br/microsoft-lanca-o-indice-de-cidadania-digital-e-incentiva-as-pessoas-a-ter-mais-empatia-online/. Acesso em Junho de 2019., 2017.

MINNESOTA/USA. University of Minnesota/USA Research, Disponível em: http://www1.umn.edu/news/news-releases/2008/UR_RELEASE_MIG_4855.html. Acesso em Junho de 2019, 2008.

PAEMAL. PHP Analysis Environment Applied to Malware Machine Learning, Disponível em: https://github.com/rewema/paemal. Acesso em Junho de 2019., 2019.

PALOALTO. The network security company. The Modern Malware Review. Analysis of New and Evasive Malware in Live Enterprise Networks, Primeira edição, 2013.

PEKTAS, A.; ACARMAN, T. Classification of Malware Families based on Runtime Behaviors. , Journal of Information Security and Applications 37 (2017) 91-100., 2017.

SANS. SANS Institute InfoSec Reading Room. Out with The Old, In with The New: Replacing Traditional Antivirus, Disponível em: https://www.sans.org/reading-room/whitepapers/analyst/old-new-replacing-traditional-antivirus-37377. Acesso em Junho de 2019., 2019.

SESHAGIRI, P.; VAZHAYIL, A.; SRIRAM, P. AMA: Static Code Analysis of Web Page for the Detection of Malicious Scripts. , Procedia Computer Science, Volume 93, 768-773, 2016.

SKYBOX. Skybox Security Vulnerability and Threat Trends Report 2018.Analysis of current vulnerabilities, exploits and threats in play., Disponível em: https://lp.skyboxsecurity.com/rs/skyboxsecurity/images/Skybox_Report_Vulnerability_Threat_Trends_18.pdf. Acesso em Junho de 2019, 2018.

SKYCURE. Skycure Mobile Threat Defense. Mobile Threat Intelligence Report Q1 2016, Disponível em:

https://www.symantec.com/content/dam/symantec/docs/reports/skycure-mobile-threat-intelligence-report-q1-2016-en.pdf. Acesso em Junho de 2019, 2016.

SOPHOS. Sophos Security made simple. Security Threat Report 2014. Smarter, Shadier, Stealthier Malware, Disponível em: https://www.sophos.com/en-us/medialibrary/pdfs/other/sophos-security-threat-report-2014.pdf. Acesso em Junho de 2019, 2014.

SYMANTEC. Symantec Reports. Internet Security Threat Report: 2001 Trends, Volume 17, Symantec Corporation., 2012.

VIRUS. VirusShare: malware files database, Disponível em: https://virusshare.com. Acesso em junho de 2019, 2019.

VIRUSSHARE. VirusShare: malware files database, Disponível em: https://virusshare.com. Acessado em Junho de 2019, 2019.

VIRUSTOTAL. Online service in order to identify malware files by main commercial antiviruses worldwide, Disponível em: https://www.virustotal.com. Acesso em Fevereiro de 2019., 2019.

WANG, Y.; QIU, Y.; THAI, T.; MOORE, K.; LIU, H.; B, Z. A two-step convolutional neural network based computer-aided detection scheme for automatically segmenting adipose tissue volume depicting on CT images., Computer Methods and Programs Biomedicine. 144:97-104, 2017.

# Modelando, com o Método ERi*c, um Aplicativo para Gerenciamento da Fila de um Restaurante

**Antonio de Padua Albuquerque Oliveira [1], Carolina dos Santos Aquino[1], Marcio Presley Farias Melo[1]**

[1] Universidade do Estado do Rio de Janeiro – UERJ
Rua São Francisco Xavier, 524 – 6º andar - Maracanã - Rio de Janeiro, Brazil

padua.uerj@gmail.com.br, aquinocarolina@ymail.com, presley.melo@gmail.com

***Abstract.*** *This paper proposes to show the application of the ERi*c method (Intentional Requirements Engineering Strategy) in modeling the requirements of a system that aims to manage the queue of a restaurant. The ERi * c method is a collaboration with Goal Oriented Requirements Engineering (GORE). An important feature of the method is the ability to reduce the complexity of iStar (i*) models used to create intentional models. For demonstration, all six steps that make up the ERi*c method were applied to the proposed case study. This article acts as a course completion project to obtain a bachelor's degree in Computer Science.*

***Resumo.*** *Este artigo propõe-se a mostrar a aplicação do método ERi*c (Engenharia de Requisitos Intencional) na modelagem dos requisitos de um sistema que visa gerenciar a fila de espera de um restaurante. O método ERi*c é uma colaboração à Engenharia de Requisitos Orientada a Metas (GORE). Uma importante característica do método está na capacidade de reduzir a complexidade dos modelos iStar (i*), utilizados para a criação de modelos intencionais. Para demonstração, foram aplicadas todas as seis etapas que compõem o método ERi*c ao estudo de caso proposto. Este artigo atua como projeto de conclusão de curso para obtenção de grau de bacharel em Ciência da Computação.*

## 1. Introdução

Um planejamento detalhado é uma questão importante no desenvolvimento de qualquer software de qualidade. Por planejamento entende-se a identificação dos objetivos do software (as metas dos atores) e o detalhamento de tempo e recursos para o encaminhamento da solução. Diante de um bom planejamento torna-se possível estimar os recursos necessários para a elaboração do sistema, definir as melhores ferramentas a serem utilizadas, bem como fazer um levantamento dos requisitos necessários.

A "Etapa de Requisitos" é responsável por 40 a 60% dos problemas encontrados nos projetos de software [Leffingwell 97]. Uma elicitação falha, pode resultar em um sistema que não atinge seus objetivos, tendo como consequência um software que foge da proposta inicial e que não atende a todas as necessidades do cliente. Por esse motivo,

a etapa de requisitos é uma fase muito importante no desenvolvimento de software, devido à necessidade de identificar possíveis problemas antes de iniciar a implementação.

Engenharia de Requisitos é o processo sistemático de levantamento, entendimento, análise, documentação e gerência de requisitos [Kotonya 98]. É o ramo da Engenharia de Software que contempla as atividades do processo de software responsável por tratar os requisitos funcionais e não funcionais de um sistema a ser construído.

Nesse contexto, a Engenharia de Requisitos Orientada a Metas (*GORE*) está preocupada com o uso de metas para elicitar, elaborar, estruturar, especificar, analisar, negociar, documentar e modificar requisitos funcionais e não funcionais [Lamsweerde 01]. A orientação a metas desempenha um papel importante no processo de Engenharia de Requisitos [Lamsweerde 98]. As metas representam a intencionalidade dos atores (pessoas e grupos organizacionais) envolvidos, e a intencionalidade espelha os interesses e as decisões desses atores. Para se aproximar mais do sucesso, sistemas de software devem atender as metas dos atores de uma organização [Oliveira 08].

## 2. Método ERi*c

O método ERi*c (Engenharia de Requisitos Intencional) é uma contribuição a Engenharia de Requisitos Orientada a Metas (*GORE*) e se baseia no conceito de intencionalidade como apresentado pelo Framework i* [Yu 95]. O método é composto por um conjunto de técnicas úteis para auxiliar a construção de modelos *iStar*. Ao término da aplicação das etapas do método ERi*c, o engenheiro produzirá os quadros e os diagramas que compõem o documento de requisitos do método.

A Figura 1 mostra uma visão geral do método ERi*c. O mnemônico ERi*c do método tem um correspondente em inglês: *IRES - Intentional RE Strategy*.

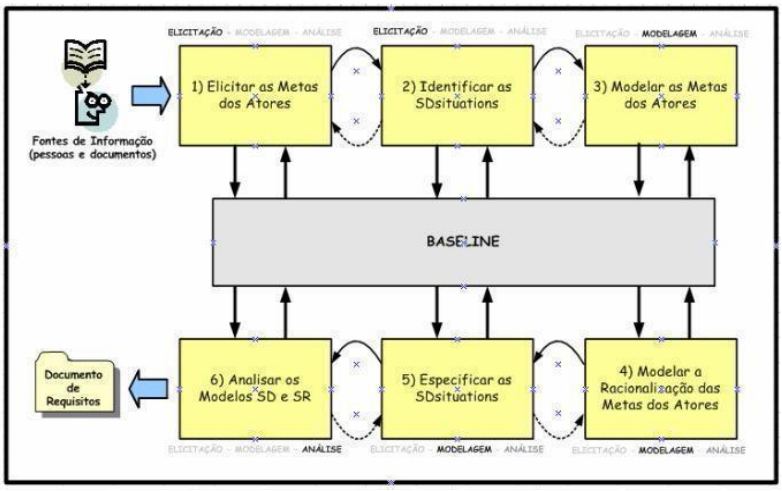



**Figura 1. Visão geral do método ERi*c [Oliveira 08]**

O método aborda as atividades de elicitação, modelagem e análise referenciadas por [Oliveira 08] como foram definidas e nomeadas primeiramente por [Leite 95], estas três atividades estão distribuídas pelas seis etapas do método. As atividades estão destacadas em negrito na Figura 1, indicando quais etapas estão preferencialmente

incluídas em cada atividade. As duas primeiras etapas do método envolvem a elicitação, na primeira etapa é feita a elicitação das metas e a identificação dos atores e, na segunda etapa, são identificadas as situações de dependência estratégica. As três etapas seguintes englobam a modelagem, na terceira etapa é elaborada a modelagem das metas dos atores, em seguida é modelada a racionalização das metas dos atores e, na quinta etapa, são especificadas as situações de dependência estratégica. Para concluir, na sexta etapa, é executada a análise dos modelos SD (dependências estratégicas – *Strategic Dependency*) e SR (razões estratégicas – *Strategic Rationale*).

O termo *baseline* [Leite 95], presente no centro do diagrama da Figura 1, indica que o método deve ser apoiado por ferramentas de software e por uma base de documentação, com a missão de facilitar o trabalho do engenheiro de requisitos no processo e também garantir, de modo automatizado, a rastreabilidade dos requisitos [Ramesh 01], além de permitir o armazenamento dos diagramas e dos documentos produzidos. O objetivo é que o método ERi*c colabore na qualidade final dos requisitos, e aperfeiçoar a qualidade final dos requisitos implica em modelos de requisitos mais completos, organizados e menos sobrecarregados.

O propósito desta seção foi apresentar uma visão geral do método ERi*c. Nas seções seguintes, descrevemos cada etapa do método, como mostrado na Figura 1, e apresentamos a modelagem do aplicativo do estudo de caso proposto.

## 3. Trabalhos Relacionados

Alguns estudos [Oliveira 08b, 17] mostram a criação de modelos intencionais utilizando o método ERi*c. Dentre estes, o artigo publicado por Oliveira, Leite, Cysneiros e Da Silva [Oliveira 17] apresenta uma ferramenta capaz de auxiliar o engenheiro de requisitos no desenvolvimento da primeira etapa do método ERi*c, usando como exemplo um sistema de pedágio (*TRC System*) para ilustrar a estratégia proposta.

Outro artigo publicado por Oliveira, Leite e Cysneiros [Oliveira 08b] apresenta todas as etapas do método ERi*c de maneira mais detalhada, através do exemplo de um sistema para apoiar a organização de conferências (*Expert Commitee*).

## 4. Modelagem ERi*c do Aplicativo do Restaurante

O sistema modelado instanciou o problema do gerenciamento e controle da fila de um restaurante típico. A fila de um restaurante, de um modo corriqueiro, provoca nos clientes ansiedade por estes não terem conhecimento de seu lugar na lista de atendimento e, por outro lado, provoca desconfiança por eles não entenderem como o número de pessoas (que é um requisito de otimização da disponibilidade do tamanho das mesas) afeta a ordem de chamada para a alocação dos clientes às mesas.

Os requisitos do “aplicativo” foram obtidos das metas elicitadas para o contexto organizacional do problema contemplando a visualização do cliente da evolução da fila do restaurante, o consumo de bebidas durante a espera e outros. O sistema aborda uma serie de problemas frequentes do cotidiano (problemas com requisitos específicos que podem ser adequadamente elicitados) como a fila de processos jurídicos, a marcação e o atendimento de consultas médicas, a fila de transplantes etecetera.

### 4.1 Elicitar as Metas dos Atores

O objetivo da primeira etapa do método é realizar a elicitação e refinamento das metas dos atores organizacionais [Oliveira 08]. Essa primeira etapa é composta por três fases: 1. Preparar LAL - Léxico Ampliado da Linguagem, 2. Definir AGFL - Metas dos Atores Vindas do Léxico e 3. Refinar as Metas dos Atores.

#### 4.1.1. Preparar LAL - Léxico Ampliado da Linguagem

O LAL descreve a linguagem do domínio do sistema e está ancorado numa ideia muito simples: Entender a linguagem do problema, sem se preocupar com a compreensão do problema [Leite 00].

Para a elaboração do LAL, o engenheiro de requisitos primeiro identifica as fontes de informação que podem ser utilizadas. São considerados todos os documentos disponíveis, e todas as pessoas que podem fornecer informações sobre o contexto organizacional objeto do novo sistema. Os seguintes passos são recomendados [Leite 00]: (i) identificar a lista de símbolos relevantes, símbolos são as palavras ou frases peculiares e mais usadas; (ii) classificar os símbolos como: sujeito (o que pratica a ação), objeto (o que sofre a ação), estado (uma situação em um dado momento) e verbo (representa uma ação); (iii) descrever os símbolos através da noção e do impacto; (iv) verificar o LAL através de inspeção; e (v) validar o LAL com os atores do Universo de Informação.

As Figuras 2, 3, 4 e 5 mostram, respectivamente, exemplos do sistema modelado pelo trabalho, símbolos do tipo sujeito, objeto, estado e verbo. As figuras foram retiradas da ferramenta C&L [C&L 18], utilizada nessa fase.

| | |
|---|---|
| **Nome:** | recepcionista |
| **Noção:** | Funcionário do restaurante que realiza o primeiro atendimento ao **cliente**. |
| **Classificação:** | sujeito |
| **Impacto(s):** | - Recepcionista aceita novos números de contato de **clientes** na **fila**.<br>- Recepcionista encerra **conta da fila** do **cliente na fila**.<br>- Recepcionista vincula **reserva** a uma **mesa**. |
| **Sinônimo(s):** | |

**Figura 2. Símbolo do LAL do tipo sujeito**

| | |
|---|---|
| **Nome:** | mesa |
| **Noção:** | Local onde **cliente** faz suas refeições. |
| **Classificação:** | objeto |
| **Impacto(s):** | - Mesa é vinculada a uma **reserva** pelo **recepcionista**.<br>- Garçom libera mesa. |
| **Sinônimo(s):** | |

**Figura 3. Símbolo do LAL do tipo objeto**

| | |
|---|---|
| **Nome:** | cliente na fila |
| **Noção:** | **cliente** que aguarda **mesa livre**. |
| **Classificação:** | estado |
| **Impacto(s):** | - **cliente** na **fila** faz **pedido**.<br>- **cliente** na **fila** acompanha situação da **fila**.<br>- **cliente** na **fila** altera **reserva**.<br>- **cliente** na **fila** paga **conta da fila**.<br>- **cliente** na **fila** sai da **fila**. |
| **Sinônimo(s):** | |

**Figura 4. Símbolo do LAL do tipo estado**

| | |
|---|---|
| **Nome:** | efetuar reserva |
| **Noção:** | **cliente** solicita entrada na **fila** de espera |
| **Classificação:** | verbo |
| **Impacto(s):** | - **cliente fora da fila** informa dados de contato.<br>- **cliente fora da fila** entra na **fila** de espera.<br>- **cliente na fila** aguarda recebimento de **alerta**.<br>- **cliente na fila** pode **sair da fila** caso desista do atendimento. |
| **Sinônimo(s):** | entrar na fila. |

**Figura 5. Símbolo do LAL do tipo verbo**

### 4.1.2. Definir AGFL – Metas dos Atores Vindas do Léxico

No LAL, os atores são identificados como sujeitos, e os impactos reportam ações desempenhadas pelo ator. A finalidade dessa atividade é identificar a motivação por trás de cada ação. Uma ação que altera ou modifica um estado é chamada de ação concreta, quando a ação proporciona uma qualidade a um estado, é chamada de ação flexível.

Para definir o *template* do método AGFL, passamos por duas subatividades: Identificar os atores e extrair as metas dos atores a partir dos impactos dos símbolos. A segunda subatividade foi realizada com o auxílio da ferramenta AGFL *Tool* [Oliveira 17].

#### 4.1.2.1. Identificar os Atores

Para identificar os atores a partir do LAL, verificamos, em primeiro, os impactos dos símbolos classificados como sujeito, pois eles praticam ações. Em segundo, analisamos os impactos dos símbolos classificados como objeto, porque esses símbolos sofrem ações praticadas por atores. Depois são analisados os impactos dos símbolos classificados com estado e verbo.

Portanto, os pretendentes naturais à posição de ator são os praticantes das ações sobre os objetos e os símbolos do tipo sujeito.

#### 4.1.2.2. Extrair as Metas dos Atores a Partir dos Impactos dos Símbolos

A finalidade dessa atividade é extrair as metas a partir dos símbolos do LAL, seguindo o *template* próprio para cada tipo de ação classificada (concreta ou flexível). De acordo com o *template* e a classificação da ação, cada impacto irá gerar uma meta em um formato pré-estabelecido [Oliveira 08]. Uma meta concreta (*goal*), quando o impacto mencionar uma ação concreta, ou uma meta flexível (*softgoal*), quando o impacto mencionar uma ação flexível.

Pela subatividade de extração das metas, é possível estabelecer a posição de cada ator como "*depender*" ou "*dependee*". Essa posição é definida, respectivamente, como: aquele que depende (*depender*) e aquele de quem se depende (*dependee*) para o alcance de uma meta.

Os Quadros 1, 2, 3, 4 e 5 mostram quatro metas concretas e uma meta flexível, extraídas a partir dos símbolos do LAL, cada uma seguindo o *template* pré-estabelecido de acordo com sua classificação. Os Quadros 1, 2 e 3 mostram metas em que o ator (*depender*) depende de outros atores (*dependee*) para atingir a meta em questão.

**Quadro 1. Extração de metas concretas de símbolo do tipo sujeito**

| | *goal* | | | ATOR |
|---|---|---|---|---|
| **-- impacto<br>resposta ao "por que?"** | **sujeito / objeto<br>LAL** | **seja** | **verbo** | **sujeito LAL** |
| **RECEPCIONISTA** | | | | |
| **recepcionista vincula reserva a uma mesa** | | | | |
| Porque **recepcionista** quer que | **mesa** | seja | **ocupada** | por **cliente** |
| Porque **cliente** quer que | **reserva** | seja | **atendida** | por **recepcionista** |

**Quadro 2. Extração de metas concretas de símbolo do tipo objeto**

| | *goal* | | | ATOR |
|---|---|---|---|---|
| **-- impacto<br>resposta ao "por que?"** | **sujeito / objeto<br>LAL** | **seja** | **verbo** | **sujeito LAL** |
| **MESA** | | | | |
| **garçom libera mesa** | | | | |
| Para que | **mesa** | seja | **alocada** | por **recepcionista** |
| Porque **recepcionista** quer que | **mesa** | seja | **ocupada** | por **cliente** |
| Porque **cliente** quer que | **reserva** | seja | **efetivada** | por **garçom** |

**Quadro 3. Extração de metas concretas de símbolo do tipo estado**

| | *goal* | | | ATOR |
|---|---|---|---|---|
| **-- impacto<br>resposta ao "por que?"** | **sujeito / objeto<br>LAL** | **seja** | **verbo** | **sujeito LAL** |
| **CLIENTE NA FILA** | | | | |
| **cliente na fila faz pedido** | | | | |
| Para que | **pedido** | seja | **servido** | por **garçom** |
| Por que **garçom** quer que | **item do cardápio** | seja | **selecionado** | por **cliente** |

**Quadro 4. Extração de meta concreta de símbolo do tipo verbo**

| | goal | | | ATOR |
|---|---|---|---|---|
| **-- impacto<br>resposta ao "por que?"** | **sujeito / objeto<br>LAL** | **seja** | **verbo** | **sujeito LAL** |
| **EFETUAR RESERVA** | | | | |
| **cliente fora da fila entra na fila de espera** | | | | |
| Para que | **reserva** | seja | **desfrutada** | por **cliente** |

**Quadro 5. Extração de meta flexível**

| | *softgoal* | | |
|---|---|---|---|
| **-- impacto<br>resposta ao "por que?"** | **Tipo<br>atributo de<br>qualidade** | **[TÓPICO]<br>sujeito / objeto<br>LAL** | **Meta concreta associada** |
| **ATUALIZAR CARDÁPIO DA FILA** | | | |
| **cliente na fila acessa cardápio da fila atualizado** | | | |
| Para que | **boas opções** | **[item do cardápio]** | **pedido** seja **consumido** por **cliente** |

### 4.1.3. Refinar as Metas dos Atores

O propósito da atividade é agrupar as metas (*goals* e *softgoals*) por ator e organizá-las em ordem cronológica [Oliveira 08]. Para tal fim, é preciso converter as metas do tipo objeto em metas do tipo sujeito, em seguida, as metas são agrupadas por ator. Por fim, elas são

ordenadas temporalmente, colocando as metas de mais longo prazo para o final, além de serem removidas as metas redundantes provenientes do processo de elicitação.

O Quadro 6 mostra as metas refinadas do ator Cliente.

**Quadro 6. Metas refinadas do ator Cliente**

| ***Depender*** | | | | | | ***Dependee*** |
|---|---|---|---|---|---|---|
| **CLIENTE** | | | | | | |
| comunicabilidade [fila] | reserva | seja | feita | | | |
| transparência [fila] | reserva | seja | cancelada | | | |
| rápida atualização [fila] | reserva | seja | cancelada | | | |
| rapidez no preparo [pedido] | pedido | seja | servido | | por | garçom |
| boas opções [item do cardápio] | pedido | seja | consumido | | | |
| flexibilidade [reserva] | reserva | seja | atendida | | por | recepcionista |
| | conta da fila | seja | quitada | | | |
| | conta da fila | seja | encerrada | | por | recepcionista |
| limpeza [mesa] | mesa | seja | ocupada | | | |

## 4.2. Identificar as Situações de Dependência Estratégica (*SDsituations*)

O objetivo da segunda etapa do método ERi*c é o refinamento e a organização das metas [Oliveira 08]. Essa etapa é composta por três subetapas: 1. Distinguir *SDsituations*, 2. Reconhecer as interdependências entre as *SDsituations* e 3. Construir o Diagrama de *SDsituations*.

Uma *SDsituation* define um bloco de elementos de dependência com intencionalidade (meta) situacional compartilhada [Oliveira 08].

### 4.2.1. Distinguir *SDsituations*

A Figura 6 exibe as *SDsituations* identificadas na modelagem do estudo de caso proposto com seus respectivos atores envolvidos.

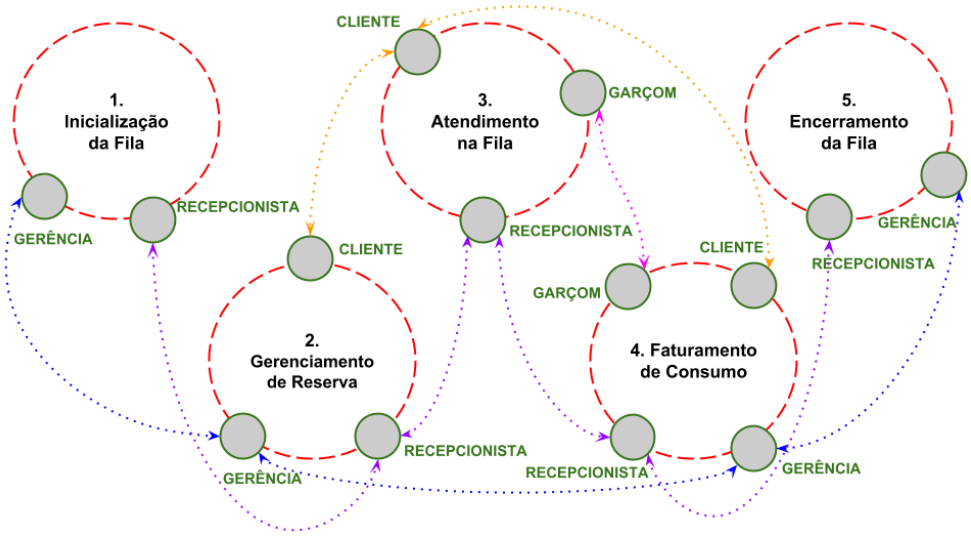



**Figura 6. *SDsituations* e atores envolvidos**

A atividade tem o propósito de identificar, em cada situação, um agrupamento de atores e suas metas em que ocorre uma dependência estratégica. Identificando como as metas estão ligadas entre si de forma a resultar em uma situação de negócio; uma situação estratégica de colaboração entre os atores; uma *SDsituation*.

No Quadro 7 estão identificadas as *SDsituations* para as metas refinadas anteriormente, na etapa 1.3.

**Quadro 7. Identificação das *SDsituations***

| ***Depender*** | ***SDsituation*** | | | | | ***Dependee*** |
|---|---|---|---|---|---|---|
| **CLIENTE** | | | | | | |
| comunicabilidade [fila] | 2 | reserva | seja | feita | | |
| transparência [fila] | 2 | reserva | seja | cancelada | | |
| rapida atualização [fila] | 2 | reserva | seja | cancelada | | |
| rapidez no preparo [pedido] | 3 | pedido | seja | servido | por | garçom |
| boas opções [item do cardápio] | 3 | pedido | seja | consumido | | |
| flexibilidade [reserva] | 2 | reserva | seja | atendida | por | recepcionista |
| | 4 | conta da fila | seja | quitada | | |
| | 4 | conta da fila | seja | encerrada | por | recepcionista |
| limpeza [mesa] | 3 | mesa | seja | ocupada | | |
| **GARCOM** | | | | | | |
| rapidez no atendimento [item do cardápio] | 3 | item do cardapio | seja | escolhido | por | cliente |
| rapidez no preparo [pedido] | 3 | pedido | seja | servido | | |
| | 3 | item consumido | seja | registrado | | |
| | 4 | item consumido | seja | cobrado | | |
| limpeza [mesa] | 3 | mesa | seja | alocada | por | recepcionista |
| **GERENCIA** | | | | | | |
| ideal acomodação [cliente] | 1 | fila | seja | organizada | por | recepcionista |
| rápida atualização [fila] | 2 | reserva | seja | excluída | | |
| transparência [fila] | 2 | reserva | seja | excluída | | |
| | 4 | conta da fila | seja | quitada | por | cliente |
| bom atendimento [fila] | 4 | cliente | seja | recompensado | | |
| | 5 | reserva | seja | atendida | por | recepcionista |
| **RECEPCIONISTA** | | | | | | |
| comunicabilidade [fila] | 2 | reserva | seja | feita | por | cliente |
| | 1 | fila | seja | autorizada | por | gerencia |
| | 4 | conta da fila | seja | cobrada | por | gerencia |
| | 4 | conta da fila | seja | transferida | | |
| | 4 | conta da fila | seja | quitada | por | cliente |
| | 4 | conta da fila | seja | encerrada | | |
| comunicabilidade [fila] | 3 | cliente | seja | notificado | | |
| limpeza [mesa] | 3 | mesa | seja | alocada | | |
| limpeza [mesa] | 3 | mesa | seja | ocupada | por | cliente |
| zerada [fila] | 5 | fila | seja | encerrada | por | gerencia |

### 4.2.2. Reconhecer as Interdependências entre as *SDsituations*

Nesta subetapa, é verificado, em cada *SDsituation*, se há situações de dependência temporal, lógica ou sequencial com outra *SDsituation*, além de analisar a presença de algum paralelismo entre as *SDsituations*.

No estudo de caso proposto, identificamos que "Inicialização da Fila" é a *SDsituation* inicial para que todas as outras *SDsituations* possam acontecer. Em seguida, "Gerenciamento de Reserva" ocorre em paralelo com "Atendimento na Fila" e elas acontecem até que "Faturamento de Consumo" seja efetivado. Por fim, "Encerramento da Fila" é a última situação de dependência estratégica para o ciclo de um dia.

### 4.2.3. Construir Diagrama de *SDsituations*

A atividade tem como propósito a construção de um diagrama que apresente as interdependências sequenciais entre as *SDsituations*. A Figura 7 mostra o diagrama de *SDsituations* do domínio estudado.

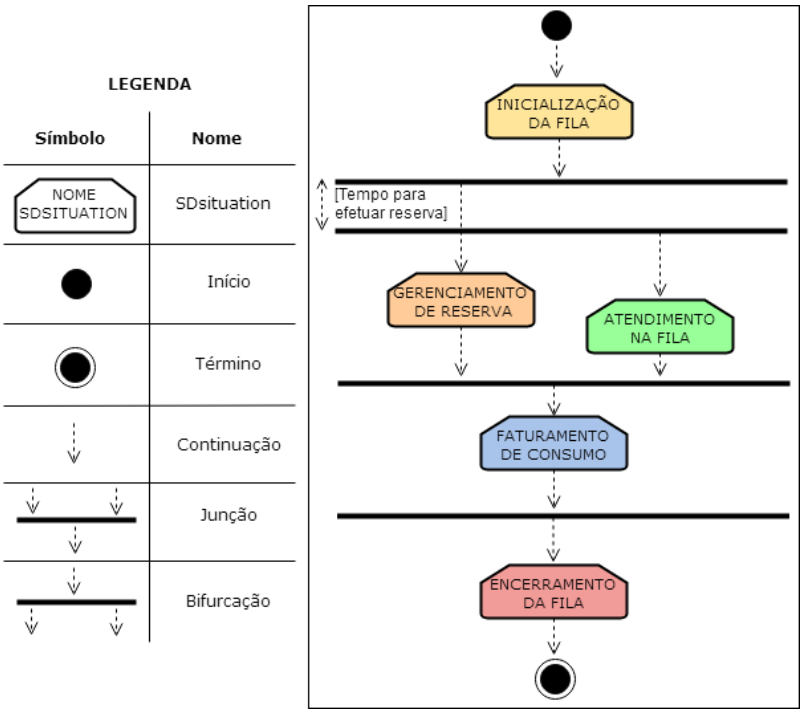



**Figura 7. Diagrama de *SDsituations***

## 4.3. Modelar as Metas dos Atores

A finalidade da terceira etapa é a modelagem e avaliação das metas dos atores [Oliveira 08]. Esse passo é dividido em duas subetapas: 3.1. Identificar Agentes, Posições e Papéis e 3.2. Criar os Painéis de Intencionalidade.

### 4.3.1. Identificar Agentes, Posições e Papéis

Em cada *SDsituation*, os atores atuam reciprocamente para atingir a meta principal da situação identificada. Para esse fim, eles podem assumir posições e exercer papéis propícios para cada *SDsituation*.

Com a separação por *SDsituations* o Engenheiro de Requisitos lida com um número reduzido de metas por vez e, além disso, elas são restritas a situação, assim ele

pode perceber com mais clareza como os atores tem algum comportamento que pode ser classificado como especialização do ator, como agente, papel ou posição [Oliveira 08].

Metas exclusivas de um ator indicam um papel interpretado pelo ator. Um ou mais papéis, geralmente exclusivos a uma única *SDsituation*, podem estar relacionados a um cargo coberto (posição) por um agente da organização, que pode surgir em mais de uma *SDsituation*.

### 4.3.2. Criar os Painéis de Intencionalidade (Diagrama IP)

Essa atividade objetiva construir um diagrama para cada *SDsituation*. O Diagrama IP é uma redução do Modelo SR do *Framework iStar*, que será visto na etapa seguinte. Nele são considerados 3 elementos: os atores, as metas (concretas e flexíveis) e as relações entre elas. O Diagrama IP é um tipo de diagrama de transição de estados que também representa os atores. Ele é um diagrama de estados por ter estados (que são as metas) conectados pelas transições [Oliveira 07].

A principal motivação da criação do Diagrama IP é a representação da intencionalidade em um único e homogêneo diagrama [Oliveira 08].

Para exemplificar o conceito de Diagrama IP, a Figura 8 mostra um dos diagramas IP do estudo de caso estudado, acompanhado da descrição de cada item desse diagrama.

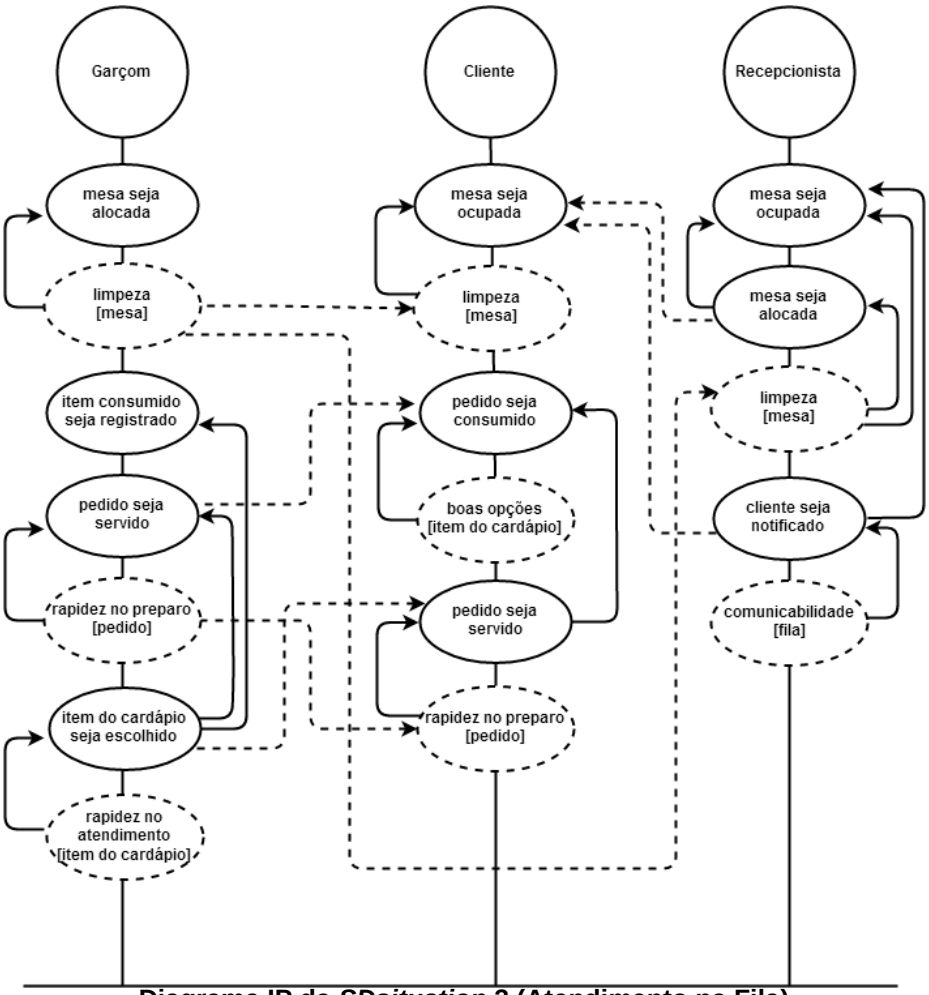



**Figura 8. Diagrama IP da *SDsituation* 3 (Atendimento na Fila)**

A Figura 8 apresenta o Diagrama IP da ***SDsituation* 3 – Atendimento na Fila**, neste diagrama é possível visualizar as relações de dependência estratégica entre três atores: Garçom, Cliente e Recepcionista.

Garçom tem como meta principal (ou meta fim) que 'mesa seja alocada', essa meta possui como atributo de qualidade a meta flexível 'limpeza [mesa]'. Ele ainda possui as metas 'item consumido seja registrado', que só pode acontecer após 'item do cardápio seja escolhido', e 'pedido seja servido', essa última possui como atributo 'rapidez no preparo [pedido]', é item de dependência para Cliente em 'pedido seja consumido', e também ocorre apenas após 'item do cardápio seja escolhido'. 'Item do cardápio seja escolhido' possui o atributo 'rapidez no atendimento [item do cardápio]', além de também ser item de dependência para Cliente.

Cliente tem como meta principal 'mesa seja ocupada', essa meta depende de duas outras metas do Recepcionista, sendo elas 'mesa seja alocada' e 'cliente seja notificado', e ainda possui como atributo 'limpeza [mesa]', essa última com dependência do Garçom. Cliente ainda possui a meta 'pedido seja consumido', que tem como atributo 'boas opções [item do cardápio]' e só ocorre após 'pedido seja servido', essa última meta possui o atributo 'rapidez no preparo [pedido]' que também depende do Garçom.

Recepcionista possui a meta principal 'mesa seja ocupada', essa meta tem como atributo 'limpeza [mesa] com dependência do garçom, e somente acontece após as demais metas que são 'mesa seja alocada', também com atributo 'limpeza [mesa]', e 'cliente seja notificado', com atributo 'comunicabilidade [fila]'.

### 4.4. Modelar a Racionalização das Metas dos Atores

O objetivo dessa etapa é construir o *rationale* dos atores. A quarta etapa do método ERi*c é formada por duas subetapas: 1. Construir Modelos SD e 2. Construir Modelos SR.

#### 4.4.1. Construir Modelos SD

As etapas do Método ERi*c estão encadeadas, portanto, para construir os Modelos SD, o engenheiro de requisitos necessita do Diagrama de *SDsituations*, elaborado na etapa 4.2.3, e dos Diagramas IP, de cada *SDsituation*, construídos na etapa 4.3.2.

O propósito da atividade é definir as dependências estratégicas entre os atores, usando os critérios do *Framework iStar* [Yu 95].

O Modelo SD é construído da seguinte maneira: Para cada *SDsituation* do Diagrama de *SDsituations*, o engenheiro de requisitos deve observar as relações de dependência assinaladas no Diagrama IP e, para cada uma delas, definir entre os quatro tipos de dependência possíveis (por meta concreta, por meta flexível, por recurso ou por tarefa) qual delas é a mais indicada e vantajosa para o "*depender*" [Oliveira 08].

Cada tipo de dependência tem uma particularidade, com relação à cooperação dos atores envolvidos. Na dependência por recurso, o *dependee* deve disponibilizar o recurso para o *depender* utilizar. Na dependência por tarefa, o *dependee* realiza uma tarefa, seguindo as instruções definidas pelo depender. Na dependência por meta concreta, o *depender* não exerce influência sobre como o *dependee* executará a meta. Na dependência por meta flexível (*softgoal*), o *depender* avalia e aceita se achar que a meta foi razoavelmente satisfeita.

Após a definição das dependências estratégicas, o Modelo SD pode ser construído. A Figura 9 ilustra o Modelo SD da ***SDsituation* 3 – Atendimento na Fil**a do estudo de caso estudado, seguido de sua descrição.

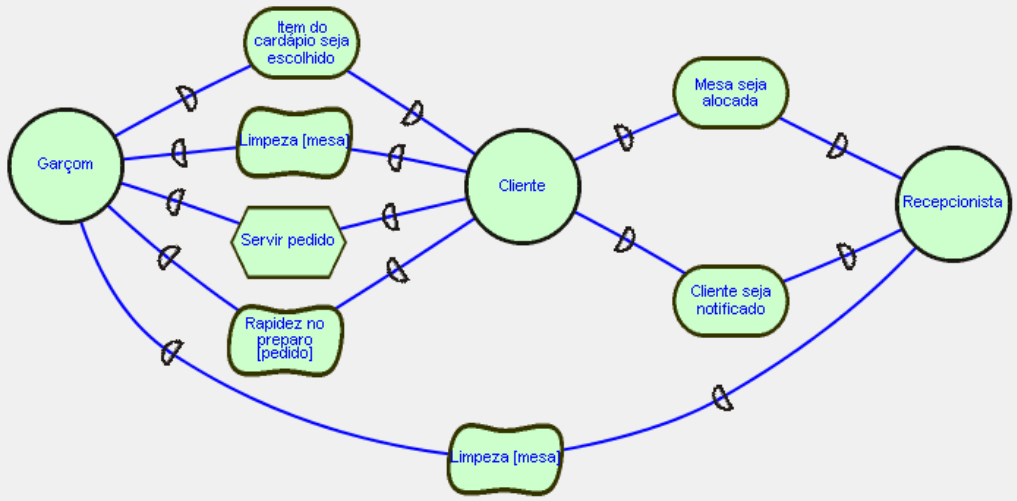



**Figura 9. Modelo SD da *SDsituation* 3 (Atendimento na Fila)**

O Diagrama IP da ***SDsituation* 3 - Atendimento na fila**, mostra quatro dependências estratégicas entre Garçom e Cliente, duas dependências entre Cliente e Recepcionista e uma entre Recepcionista e Garçom. As dependências foram representadas da seguinte maneira no Modelo SD, como visto na Figura 9:

Entre Garçom e Cliente:

- por meta concreta em 'item do cardápio seja escolhido', porque o Garçom não tem interferência na escolha dos itens pelo Cliente;
- por meta flexível em 'limpeza [mesa]', por ser atributo de qualidade;
- por tarefa em 'servir pedido', porque Cliente escolhe como será servido o pedido;
- por meta flexível em 'rapidez no preparo [pedido]', por ser atributo de qualidade.

Entre Cliente e Recepcionista:

- por meta concreta em 'mesa seja alocada', porque o Recepcionista pode realizar essa meta sem interferência do Cliente;
- por meta concreta em 'cliente seja notificado', porque o Cliente não tem interferência sobre como o Recepcionista realizará essa meta.

Entre Recepcionista e Garçom:

- por meta flexível em 'limpeza [mesa]', por ser atributo de qualidade.

### 4.4.2. Construir Modelos SR

O Diagrama IP, o Modelo SD e o Diagrama de *SDsituations* elaborados nas etapas anteriores do Método ERi*c, são itens fundamentais para elaboração do Modelo SR.

Primeiramente, posiciona-se todas as metas, concretas e flexíveis, que aparecem no eixo de cada ator do Diagrama IP dentro da linha limite do ator no Modelo SR. Após esse passo, é definida uma tarefa como meio para atingir cada meta concreta. Em seguida as metas são associadas à meta fim no Diagrama IP como decomposição de tarefa, ou

seja, os passos necessários para alcançar determinada tarefa. Ainda são alocadas como tarefas ou recursos as metas próprias do ator [Oliveira 08].

As associações representadas por uma linha com traço no Modelo SR indicam as decomposições de tarefas, e as associações representadas por uma seta indicam uma relação meio-fim, ou seja, a tarefa necessária para se alcançar a meta fim. A Figura 10 mostra o Modelo SR da ***SDsituation* 3 – Atendimento na Fila**.

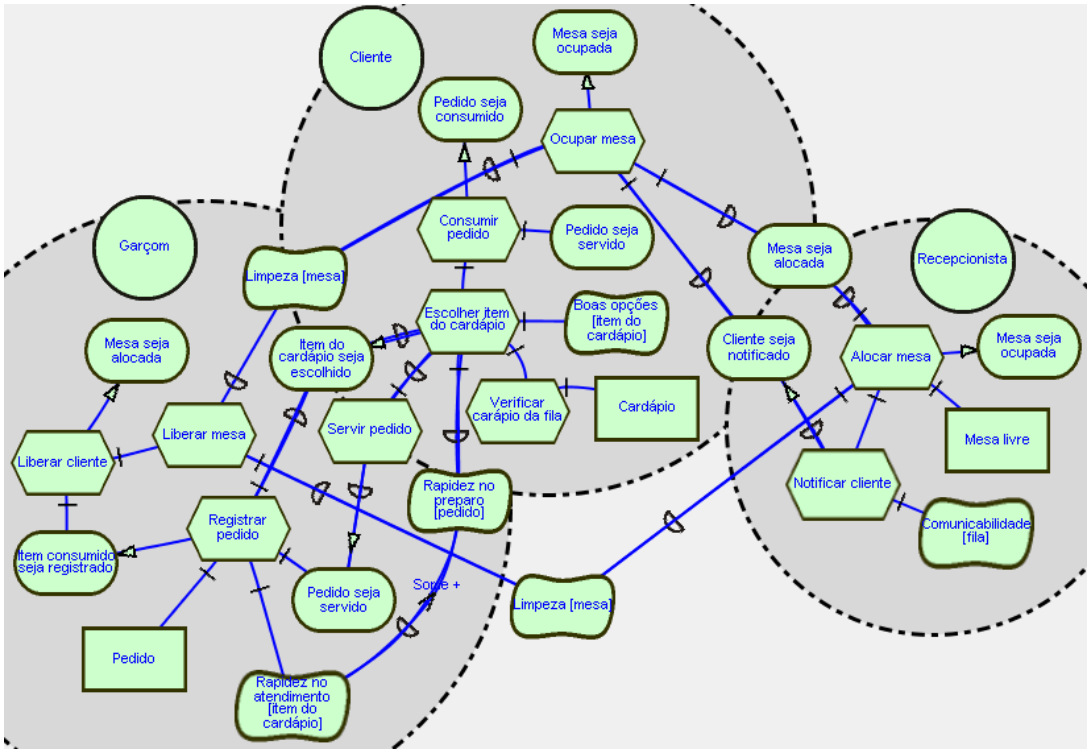



**Figura 10. Modelo SR da *SDsituation* 3 (Atendimento na Fila)**

## 4.5. Especificar as *SDsituations*

A quinta etapa é composta por uma única subetapa: 1. Descrever as Situações de Dependência Estratégica. Seu objetivo, como o próprio nome indica, é a especificação das situações de dependência estratégica [Oliveira 08].

### 4.5.1. Descrever as Situações de Dependência Estratégica

A quinta etapa do método ERi*c consiste em descrever as *SDsituations* por meio de cenários. Para realizar essa tarefa, o engenheiro de requisitos utiliza os itens produzidos nas etapas anteriores, sendo eles o Diagrama de *SDsituations* e o Modelo SR de cada *SDsituation*.

O Quadro 8 apresenta o exemplo da especificação, em forma de cenário, da ***SDsituation* 3 – Atendimento na Fila**.

**Quadro 8. Descrição da *SDsituation* Atendimento na Fila**

**Título:** ATENDIMENTO NA FILA
**Objetivo:** **Cliente seja atendido**
**Contexto:**
**Localização Geográfica:** Restaurante
**Localização temporal:** Após efetuar reserva, enquanto aguarda mesa livre
**Precondição:** Reserva foi efetuada
**Recursos:** Pedido, cardápio, mesa livre
**Atores:** Cliente, Garçom e Recepcionista
**Episódios:** Cliente faz pedido
Cliente questiona sobre item do cardápio
Garçom responde sobre item do cardápio
Cliente escolhe item do cardápio
Garçom atende cliente
Garçom anota pedido
Garçom serve pedido
**Restrição:** Para Recepcionista alocar cliente a uma mesa, a mesa precisa estar limpa.
**Exceção:** Não há.
**Metas flexíveis:** Comunicabilidade [fila], boas opções [item do cardápio], rapidez no atendimento [item do cardápio], limpeza [mesa], rapidez no preparo [pedido]

## 4.6. Analisar a Racionalização das Metas dos Atores

A sexta e última etapa do Método ERi*c tem como objetivo a verificação dos modelos *iStar* para a melhoria da qualidade [Oliveira 08], com a finalidade de prevenir defeitos ao invés de detectá-los.

Essa última etapa é dividida em três subetapas: 1. Identificar Estruturas Canônicas, 2. Aplicar o *Framework* de Perguntas e 3. Verificar as Perguntas Respondidas.

### 4.6.1. Identificar Estruturas Canônicas (*SRconstructs*)

Um *SRconstruct* possui um fim que será alcançado por um ou mais meios, estes meios são as tarefas. Para não haver repetição, não são consideradas as metas concretas subcomponentes da decomposição de tarefas, porque estas são o elemento principal de análise de outro *SRconstruct*.

Para a ***SDsituation* 3 – Atendimento na Fila**, os *SRconstructs* existentes estão identificados abaixo, organizados por ator:

- Cliente: mesa seja ocupada, pedido seja consumido, item do cardápio seja escolhido.
- Recepcionista: mesa seja ocupada, cliente seja notificado.
- Garçom: mesa seja alocada, item consumido seja registrado, pedido seja servido.

Totalizando para o exemplo oito *SRconstructs*.

### 4.6.2. Aplicar *Framework* de Perguntas

Esta subetapa consiste em dividir os diagramas i* em problemas menores, para que se faça uma análise minuciosa de cada parte do diagrama. É aplicado então um *framework* de perguntas para cada *SDsituation* e para cada *SRconstruct* da *SDsituation*.

Para melhor distribuição na página, o *framework* de perguntas aplicado à ***SDsituation* 3 - Atendimento na Fila** foi separado em três partes. O Quadro 9a contém as questões externas e os Quadros 9b e 9c contém as questões internas. O *framework* de perguntas do *SRconstruct* Mesa seja Alocada foi separado em duas partes pelo mesmo motivo, o Quadro 10a contém as questões externas e 10b contém as questões internas.

**Quadro 9a. Diagnóstico da *SDsituation* 3 Atendimento na Fila – Questões Externas**

**DIAGNÓSTICO DA SDSITUATION:**

***SDsituation*: "Atendimento na fila"**

I. QUESTÕES EXTERNAS

1. Quem mais poderia colaborar com o "**Recepcionista**" para atingir "**Cliente seja atendido**"? Quanto ele pode colaborar? (completamente ou parcialmente)
   - Garçom colabora parcialmente.
2. Por que o "**Garçom**" colabora com o "**Recepcionista**" para atingir "**Cliente seja atendido**"?
   - Para agilizar o atendimento de clientes na fila.
3. Quais *SDsituations* acontecem antes de "**Atendimento na Fila**"?
   - As *SDsituations*: Inicialização da Fila e Gerenciamento de Reserva.
4. Que problemas com as *SDsituations* anteriores podem ser identificados para atingir "**Cliente seja atendido**"?
   - Gerência demora a autorizar a inicialização da fila.
   - Dificuldade no manuseio do aplicativo.
5. E se o "**Garçom**" não puder colaborar no "**Atendimento na Fila**"?
   - Garçom deve receber o devido treinamento.

**Quadro 9b. Diagnóstico da *SDsituation* 3 Atendimento na Fila – Questões Internas**

II. QUESTÕES INTERNAS

6. Quais são os problemas dentro do "**Atendimento na Fila**"? Que tipos de problemas (precisão, deficiências, ambiguidades ou omissões) são identificados em "**Cliente seja atendido**"?
   - Travamento/lentidão do sistema
   - Recepcionistas e garçons sem o treinamento adequado.
   - Falta de comunicabilidade com o cliente.

RELAÇÃO **GARÇOM - CLIENTE** (*DEPENDER-DEPENDEE*)

7. Que detalhes o "**garçom**" necessita?
   a) Caso: dependência de recurso - não se aplica.
   b) Caso: dependência de meta concreta - Quais são os problemas que o "**item do cardápio seja escolhido**" tem para ser alcançada pelo "**cliente**"? (tempo, habilidade) Quando? Como? Quanto?
   - Não ter disponível todas as opções do cardápio.

   c) Caso: dependência de meta flexível - não se aplica.
   d) Caso: dependência de tarefa – não se aplica.

RELAÇÃO **RECEPCIONISTA - GARÇOM** (*DEPENDER-DEPENDEE*)

8. Que detalhes o "**recepcionista**" necessita?
   a) Caso: dependência de recurso – não se aplica.
   b) Caso: dependência de meta concreta – não se aplica.
   c) Caso: dependência de meta flexível - Quais são os problemas que o "**limpeza [mesa]**" tem para ser razoavelmente satisfeita pelo "**garçom**"? (capacidade) Existe "**limpeza [mesa]**" na conclusão de "**cliente seja atendido**"? Por que? Quem está demandando pela meta flexível?
   - Dificuldade para liberar a mesa no aplicativo.

   d) Caso: dependência de tarefa – não se aplica.

**Quadro 9c. Continuação do Diagnóstico da *SDsituation* 3 Atendimento na Fila – Questões Internas**

RELAÇÃO **CLIENTE - RECEPCIONISTA** (*DEPENDER-DEPENDEE*)

9. Que detalhes o "**cliente**" necessita?
   a) Caso: dependência de recurso – não se aplica.
   b) Caso: dependência de meta concreta - Quais são os problemas que o "**mesa seja alocada**" tem para ser alcançada pelo "**recepcionista**"? (tempo, habilidade) Quando? Como? Quanto?
   - Mesa não estar em condições apresentáveis.
   - Travamento/lentidão do sistema.
   - Recepcionista com dificuldade em manusear o aplicativo.

   Quais são os problemas que o "**cliente seja notificado**" tem para ser alcançada pelo "**recepcionista**"? (tempo, habilidade) Quando? Como? Quanto?
   - Falta de comunicabilidade com o cliente.

   c) Caso: dependência de meta flexível – não se aplica.
   d) Caso: dependência de tarefa – não se aplica.

RELAÇÃO **CLIENTE - GARÇOM** (*DEPENDER-DEPENDEE*)

10. Que detalhes o "**cliente**" necessita?
    a) Caso: dependência de recurso – não se aplica.
    b) Caso: dependência de meta concreta – não se aplica.
    c) Caso: dependência de meta flexível - Quais são os problemas que o "**limpeza [mesa]**" tem para ser razoavelmente satisfeita pelo "**garçom**"? (capacidade) Existe "**limpeza [mesa]**" na conclusão de "**cliente seja atendido**"? Porque? Quem está demandando pela meta flexível?
    - Dificuldade para liberar a mesa no aplicativo.
    - A contribuição é manter a mesa bem apresentada.

    Quais são os problemas que o "**rapidez no preparo [pedido]**" tem para ser razoavelmente satisfeita pelo "**garçom**"? (capacidade) Existe "**rapidez no preparo [pedido]**" na conclusão de "**cliente seja atendido**"? Porque? Quem está demandando pela meta flexível?
    - Demora para preparar o pedido.

    d) Caso: dependência de tarefa – O "**garçom**" recebeu as orientações de como fazer "**servir pedido**"? Pode o "**garçom**" realizar a tarefa? (tempo, habilidade)
    - O cliente informa ao garçom como deseja que o pedido seja servido.
    - Garçom deve receber treinamento de como atender clientes.
11. Qual dependência possui a maior responsabilidade para atingir **"cliente seja atendido"**?
    - A meta concreta **"mesa seja alocada".**

    Por que?
    - Porque o cliente depende enormemente dessa meta para ser atendido.

**Quadro 10a. Diagnóstico do *SRconstruct* Mesa seja Alocada – Questões Externas**

**DIAGNÓSTICO *SRCONSTRUCT*:**

***SRconstruct*: "mesa seja alocada"**

I. QUESTÕES EXTERNAS

12. Quem mais tem como meta "**mesa seja alocada**"?
    - Ninguém.
13. Quais são as alternativas possíveis para que "**mesa seja alocada**" seja atingida? Por quê?
    - Mesa será alocada via aplicativo.
14. Quais são os elementos de dependência dos *dependees*?
    - Meta concreta: Item consumido seja registrado. Tarefa: liberar mesa.
15. Que tipos de problemas (precisão, deficiências, ambiguidades ou omissões) podem ser vislumbrados? E se os recursos ficarem indisponíveis? Como evitar tais problemas?
    - Pedidos consumidos que ainda não foram registrados.
16. E se a "**mesa seja alocada**" for partilhada com outro ator?
    Não se aplica.
17. Que outro construto depende dessa meta? Por quê? Quanto?
    - Nenhum.

**Quadro 10b. Diagnóstico do *SRconstruct* Mesa seja Alocada – Questões Internas**

| |
|---|
| II.QUESTÕES INTERNAS (PARA CADA TAREFA MEIO)<br>18. Quais são os problemas com a tarefa "**liberar cliente**"? Por quê?<br>● Dificuldade do garçom em liberar clientes pelo aplicativo.<br>19. Para a tarefa "**liberar cliente**" que componentes são necessários para atingir "**mesa seja alocada**"?<br>● Tarefa: Liberar Mesa. Meta concreta: Item consumido seja registrado.<br>a) Caso: recurso – não se aplica.<br>b) Caso: meta flexível – não se aplica.<br>c) Caso: subTarefa – Pode o ator "**garçom**" fazer "**liberar mesa**"? (tempo, habilidade)<br>● Sim, desde que a mesa esteja limpa.<br>20. Existe algum detalhe omitido na operacionalização? De que tipo? Por quê? Como? Quanto?<br>● O garçom precisa estar atento na liberação da mesa, para que outro cliente seja acomodado rapidamente.<br>21. Há falta de algum recurso? De que tipo? E se o recurso não estiver disponível?<br>● Não há. |

### 4.6.3. Verificar as Perguntas Respondidas

A partir dos diagnósticos aplicados na subetapa anterior, é possível elaborar o Quadro de Metas x Problemas. O Quadro consiste em identificar quais metas são atingidas pelos problemas observados nas *SDsituations* e *SRconstructs*. Com o Quadro em mãos é possível visualizar mais facilmente deficiências do sistema e problemas do negócio, podendo assim reparar esses erros, antes de desenvolver a aplicação.

O Quadro 11 é a legenda de identificação das metas do Quadro 12. O Quadro 12 mostra as Metas x Problemas elaborado para a ***SDsituation* 3 – Atendimento na Fila**, a partir das perguntas respondidas no passo anterior. As metas concretas estão numeradas e as metas flexíveis estão identificadas por letras.

**Quadro 11. Legenda para identificação das metas no Quadro Metas x Problemas**

| # Metas | Nomes das Metas (*SDsituation* 3 Atendimento na Fila) |
|---|---|
| 1 | Mesa seja ocupada |
| 2 | Pedido seja consumido |
| 3 | Item do cardápio seja escolhido |
| **a** | Boas opções [item do cardápio] |
| 4 | Mesa seja ocupada |
| 5 | Cliente seja notificado |
| **b** | Comunicabilidade [fila] |
| 6 | Mesa seja alocada |
| 7 | Item consumido seja registrado |
| 8 | Pedido seja servido |
| **c** | Rapidez no atendimento [item do cardápio] |
| **d** | Rapidez no preparo [pedido] |
| **e** | Limpeza [mesa] |

**Quadro 12. Quadro Metas x Problemas da *SDsituation* 3 Atendimento na Fila**

| **Problemas** **Metas →** | **1** | **2** | **3** | **a** | **4** | **5** | **b** | **6** | **7** | **8** | **c** | **d** | **e** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (1) Interface que dificulte o manuseio | x | | | | x | x | x | x | | | x | | |
| (2) Travamento/lentidão do sistema | x | | | | x | x | x | x | | | x | | |
| (3) Dificuldade do garçom ou recepcionista no manuseio do aplicativo | x | | | | x | x | x | x | | | x | | |
| (4) Gerência demora a autorizar a inicialização da fila | x | | | | x | | | | | | | | |
| (5) Recepcionistas e garçons sem o treinamento adequado | | | | | | | | | x | x | x | x | x |
| (6) Falta de comunicabilidade com o cliente | x | | | | x | x | x | | | | | | |
| (7) Não ter disponível todas as opções do cardápio | | | x | x | | | | | | x | | | |
| (8) Demora em servir o pedido | | x | | | | | | | | x | | | |
| (9) Demora em liberar mesas | x | | | | x | | | x | | | | | |
| (10) Garçom troca ou toma nota dos pedidos erradamente | | x | | | | | | | | x | | | |
| (11) Garçom não registra pedidos consumidos ou registra erradamente | | | | | | | | | x | | | | |
| (12) Falta, em estoque, do item do cardápio escolhido ou dos ingredientes necessários | | | x | x | | | | | | | x | | |

## 5. Conclusões

O objetivo do artigo foi apresentar a aplicação do método Engenharia de Requisitos Intencional (ERi*c). Para demonstração e melhor entendimento do método, aplicamos todas as etapas do método ERi*c a um sistema que tem como objetivo o gerenciamento da fila de espera de um restaurante. Mostramos também neste artigo, os diagramas e quadros desenvolvidos ao longo do processo.

A elaboração de modelos i* pode ser muito complexa [Pastor 11], por isso o método ERi*c surgiu como uma alternativa para simplificar a produção desses modelos dividindo seu desenvolvimento em etapas menos elaboradas. O propósito foi de concluir a última etapa do método com os modelos i* prontos e com uma análise detalhada das possíveis deficiências da aplicação desejada, sejam elas falhas do sistema em si ou problemas do negócio. Sendo possível então tomar alguma medida para reparar tais problemas, antes que atinjam de fato a aplicação durante a implementação ou produção do sistema.

Para exemplificar, analisando os dados obtidos no Quadro Metas x Problemas da ***SDsituation* 3 – Atendimento na Fila** (Quadro 12), é possível verificar as metas com maior incidência de problemas e os problemas que atingem mais metas, ambos em destaque no Quadro 12. As metas 1 e 4, são atingidas por seis problemas cada. O problema 5 atinge cinco metas, os problemas 1, 2 e 3 atingem seis metas e o problema 6 atinge quatro metas cada. Nota-se que os problemas 1, 2 e 3 são problemas que atingem o sistema, já os problemas 5 e 6 são problemas que afetam o negócio. O levantamento dos problemas com maior incidência e das metas mais atingidas em todas as *SDsituations* são

importantes, pois permitem ao engenheiro de requisitos saber quais áreas do sistema devem ter a correção priorizada, e para que, nesse caso, o restaurante busque uma solução para os problemas de negócio identificados, antes que afetem o atendimento.

Analisando as demais etapas do método notamos a importância de um amplo conhecimento do estudo de caso abordado para elaboração do léxico ampliado da linguagem na primeira etapa, visto que o LAL é a base para o desenvolvimento das etapas seguintes. A ferramenta AGFL *Tool* [Oliveira 17] nos ajudou a extrair as metas dos atores a partir dos impactos dos símbolos, tornando mais ágil o refinamento das metas dos atores. As *SDsituations* identificadas na segunda etapa ajudaram a melhor entender e dividir o funcionamento do sistema em blocos com mesma intencionalidade. Podendo assim agrupar as metas de acordo com os blocos identificados. Na terceira etapa foram elaborados os Diagramas IP, para desenvolvê-los utilizamos os quadros e diagramas produzidos nas etapas anteriores. O Diagrama IP nos permitiu visualizar melhor as relações de dependência estratégica entre as metas de cada ator. Os modelos SD e SR foram construídos na quarta etapa do método, o modelo SD direcionou a construção do modelo SR que é bastante complexo, o que torna seu desenvolvimento mais trabalhoso. Como o objetivo do método ERi*c é auxiliar a construção de modelos i*, nesta etapa o próprio método fornece um passo a passo detalhado para construir os modelos SD e SR, utilizando como base os diagramas elaborados nas etapas anteriores. Ao especificar as *SDsituations* na quinta etapa, temos uma descrição detalhada que facilita o entendimento de cada *SDsituation*.

A análise do quadro Metas x Problemas, obtido na última etapa do método, mostra sua relevância na verificação das falhas do sistema. Essa análise também enfatiza a importância das etapas anteriores, pois os diagramas produzidos são necessários para elaboração do quadro final e para compor o documento de requisitos. Ressaltando assim a importância de todas as etapas do método ERi*c. Durante a aplicação do método, foi possível observar também uma menor complexidade e maior clareza na elaboração dos modelos i* [Pastor 11] seguindo os *templates* oferecidos pelo método. Concluímos assim que os resultados alcançados com a modelagem do aplicativo do estudo de caso apresentado, mostram que o método ERi*c é eficiente e consegue atingir o objetivo a que se propõe.

Oliveira, em [Oliveira 08b], fornece um guia mais detalhado das etapas do método ERi*c, e foi uma importante fonte para o artigo corrente.

Para compor o projeto de conclusão de curso modelamos os quadros e diagramas de todas as *SDsituations* do estudo de caso proposto. Contudo, apenas alguns deles foram apresentados no artigo devido ao limite de páginas. Nosso trabalho oferece como contribuição, a descrição de um novo exemplo da aplicação do método ERi*c, bem como um conjunto de quadros e diagramas úteis, capazes de auxiliar em uma futura implementação do sistema de fila de espera transparente em um restaurante. Todos os quadros e diagramas desenvolvidos e não presentes neste artigo estão disponíveis no repositório *ERicMethod* do *GitHub* [ERicMethod 19].

## Referências:

[C&L 18] Cenários e Léxicos - PUC-Rio - Disponível em: http://pes.inf.puc-rio.br/cel/. Acessado em: Janeiro/2018.

[ERicMethod 19] Repositório disponível em: https://github.com/ERicMethod Acessado em: Abril/2019.

[Kotonya 98] Kotonya, G., & Sommerville, I. (1998). *Requirements engineering: processes and techniques*. Wiley Publishing.

[Lamsweerde 98] Van Lamsweerde, A., Darimont, R., & Letier, E. (1998). Managing conflicts in goal-driven requirements engineering. *IEEE transactions on Software engineering*, *24*(11), 908-926.

[Lamsweerde 01] Van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on* (pp. 249-262). IEEE.

[Leffingwell 97] Leffingwell, D. (1997). Calculating your return on investment from more effective requirements management. *American Programmer*, *10*(4), 13-16.

[Leite 95] Leite, J. C. S. D. P., & Oliveira, A. D. P. A. (1995). A client oriented requirements baseline. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on* (pp. 108-115). IEEE.

[Leite 00] Leite, J. C. S. D. P., Hadad, G. D., Doorn, J. H., & Kaplan, G. N. (2000). A scenario construction process. *Requirements Engineering*, *5*(1), 38-61.

[Oliveira 07] Oliveira, A. D. P. A., Leite, J. C. S. D. P., Cysneiros, L. M., & Cappelli, C. (2007). Eliciting multi-agent systems intentionality: from language extended lexicon to i* models. In *Chilean Society of Computer Science, 2007. SCCC'07. XXVI International Conference of the* (pp. 40-49). IEEE.

[Oliveira 08] Oliveira, A. D. P. A. (2008). *Intentional Requirements Engineering: A Method for Requirements Elicitation, Modeling, and Analysis. 261 p* (Doctoral dissertation, Doctoral Thesis–Computer Science Department, PUC-Rio-Rio de Janeiro).

[Oliveira 08b] Oliveira, A. D. P. A., Leite, J. C. S. D. P., & Cysneiros, L. M. (2008). Método ERi* c-Engenharia de Requisitos Intencional. In *WER*.

[Oliveira 17] Oliveira, A. D. P. A., Leite, J. C. S. D. P., Cysneiros, L. M., & Da Silva, W. G. S. (2017). Eliciting Goals and Softgoals-How to Perceive the Intentionality at the Beginning of the Journey. In iStar (pp. 31-36).

[Pastor 11] Pastor, O., Estrada, H., & Martınez, A. (2011). The strengths and weaknesses of the i* framework: an experimental evaluation. *Social Modeling for Requirements Engineering. Cooperative Information Systems series. MIT Press, Cambridge*.

[Ramesh 01] Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. *IEEE transactions on software engineering*, (1), 58-93.

[Yu 95] Yu, E. (2011). Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, *11*, 2011.

# Sistema de voto eletrônico utilizando a blockchain

**Henrique Niwa, Celso Mendes**

[1]Instituto Nacional de Pesquisas Espaciais - INPE
Laboratório Associado de Computação e Matemática Aplicada - LABAC
Av. dos Astronautas, 1758, Jardim da Granja, CEP:12227-010
São José dos Campos - SP - Brasil

{henrique.niwa,celso}@inpe.br

***Resumo.*** *Este trabalho mostra o estudo e implementação de um sistema de voto eletrônico, utilizando-se de blockchain, um banco de dados descentralizado e criptografado. O sistema proposto, além de oferecer ainda mais segurança ao processo de votação, permitiria uma completa auditoria na eleição, tanto do código fonte utilizado quanto da base de dados, cada eleitor poderia verificar o seu próprio voto. O desafio envolvendo computação de alto desempenho foi distribuir, simular o processo eleitoral e apurar a votação na escala de dezenas de milhões de eleitores, assim como o modelo atual em vigência no Brasil, incluindo criptografia nas transações e total transparência na apuração.*

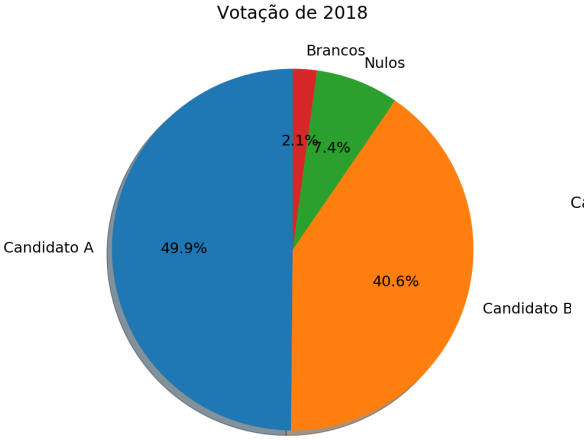
***Abstract.*** *This paper shows the study and implementation of an electronic voting system using blockchain, a decentralized and encrypted database. The proposed system, in addition to offering even more security to the voting process, would allow a complete audit of the election of both the source code used and the database, each voter could verify his own vote. The challenge involving high-performance computing was to distribute, simulate the electoral process, and to count the voting of tens of millions of voters, with a performance in the same level of the current model in place in Brazil, including transaction encryption and full transparency in polling.*

## 1. Introdução

Existem diferentes meios de votação pelo mundo, o mais comum e simples é o que se utiliza de cédulas de votação, um processo que consiste no eleitor ir a um centro designado, criar a marcação de sua preferência sendo observado por um grupo de auditores, mas com o voto ainda secreto e depositar sua ficha de papel em uma urna. Esses votos então são agregados de todos os diferentes centros de votação e posteriormente validados e contabilizados se utilizando de métodos manuais e automáticos. Para uma pequena quantidade de votantes é um meio simples e razoavelmente rápido de votar e contar, porém para grandes populações há um grande trabalho a ser feito e portanto levam-se dias para finalizar o evento. Também existe o risco de que contagens erradas, fraude nas cédulas e urnas de votação e ausência de eleitores atrapalhe e/ou mude o resultado. No Brasil temos um sistema de voto onde se utilizam urnas eletrônicas, que fazem a contagem e contabilidade local dos votos, ainda assim é necessário que as unidades de memória de cada máquina sejam enviadas para um local central e efetuada a leitura e agregação de votos, essas unidades de memória e as máquinas em si necessitam de segurança para que os dados não sejam modificados por partes maliciosas interessadas.

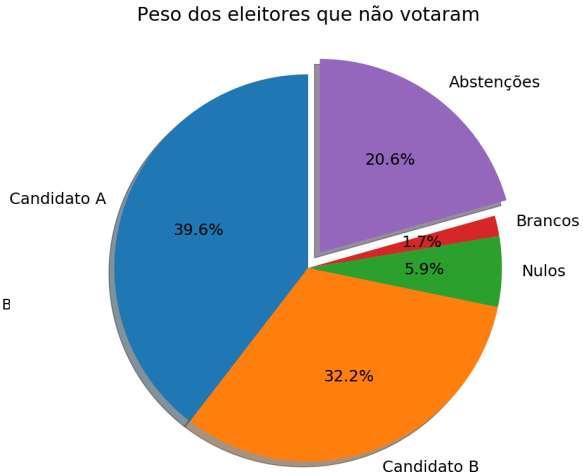
Para testes de performance do sistema foram levados em conta os números de eleitores do Brasil, em 2018[1] eram 146 milhões de eleitores esperados, porém na votação final foram contabilizados 116 milhões[2](figura 1), havendo 11 milhões de brancos e nulos, portanto quase 20 milhões de abstenções. Em 2018 as eleições[3] ocorreram em 7 de outubro, o prazo para troca[4] de zona eleitoral em 9 de maio e o pedido para voto em trânsito[5] em 23 de agosto, ou seja a população teve menos de 1 mês e meio para poder requisitar o direito ao voto sem estar em sua zona eleitoral de registro ou ainda 5 meses para poder votar nos representantes de sua localidade atual. Com o sistema proposto a localidade dos votos é irrelevante pois o eleitor poderá votar em qualquer dispositivo ou ainda em qualquer zona eleitoral. Essa desburocratização do voto pode mudar o cenário das eleições de acordo com a figura 2 a parcela de eleitores que poderia ser beneficiada é significativa com 20% do total. E o tempo de apuração em 2018 para o segundo turno foi iniciado a partir das 19h até a meia noite, 5 horas no total[6]. A suposição de que todas as abstenções seriam suprimidas com uma votação eletrônica baseada em blockchain é uma visão otimista, devendo haver uma coordenação entre as áreas da sociologia e modificação das regras eleitorais para tanto.

**Figura 1. Votos válidos de 2018.**




**Fonte: Jornal Gazeta do Povo[1]**

**Figura 2. Parcela de eleitores que poderiam votar no sistema proposto.**




**Fonte: CNM - Confederação Nacional de Municípios[2]**

[1] https://www.cnm.org.br/cms/biblioteca/Eleitorado-2018.pdf

[2] https://especiais.gazetadopovo.com.br/eleicoes/2018/resultados/brasil-2turno-presidente/

[3] https://g1.globo.com/politica/eleicoes/2018/noticia/eleicoes-2018-datas.ghtml

[4] https://exame.abril.com.br/brasil/prazo-para-pedir-ou-transferir-titulo-eleitoral-

[5] https://g1.globo.com/politica/eleicoes/2018/noticia/2018/08/22/termina-nesta-quinta-feira-prazo-para-eleitor-pedir-voto-em-transito.ghtml

[6] https://g1.globo.com/politica/eleicoes/2018/apuracao/presidente.ghtml

## 2. Metodologia

O sistema implementado buscou alguns objetivos:

1. Anonimato
2. Segurança
3. Imutabilidade
4. Acesso Remoto
5. Verificável pelo usuário
6. Auditável
7. Código aberto

O anonimato foi garantido pelo fato dos usuários terem apenas informações das chaves públicas (através dos endereços no *blockchain*) registradas no banco de dados. A segurança se teve ao utilizar criptografia estabelecida e implementações já testadas. Imutabilidade vem da própria natureza do *blockchain*, onde cada novo bloco aumenta ainda mais a segurança dos dados. Acesso remoto se teve pelo uso de uma requisição HTTP aos serviços, qualquer cliente pode ser escrito que possua uma biblioteca de requisições web. Podem ser criados clientes para computadores desktop, aplicativos para Android e iOS, navegadores web. Esta requisição utiliza JSON e poderia ser feita utilizando HTTPS assim evitando que os dados da rede sejam interceptados e se crie uma associação da chave pública com um endereço da internet e consequentemente com uma pessoa física. Verificação através do id da transação, cada voto irá retornar um identificador que irá permitir saber o destino do voto, quantidades e data. Auditabilidade completa, no final da divulgação dos resultados, é distribuído o banco de dados com todos os votos, todos poderão ver os votos, a associação entre um eleitor e sua identidade nos registros só poderá ser feito por cada um. Ainda enquanto a votação estiver sendo realizada, um nó pode ser gerido por uma auditoria, que poderá conferir em tempo real os votos. O uso de código aberto foi primordial, pois a pesquisa utilizou de verba pública, também houve cuidado ao se escolher a licença de *copyright*, pois o código original do *bitcoin* é uma licença MIT, quer dizer que o código pode ser modificado e utilizado como quiser, sem necessitar de divulgação. A licença utilizada foi a GPLv3, ela dita que qualquer modificação deve ter seus fontes divulgados.

Foi utilizado containers docker para criação de ambientes de compilação e testes, isto foi de acordo com a licença, pois segundo a GPLv3 não se deve apenas divulgar o código fonte, mas também todo o necessário para que seja feita a compilação do mesmo. Não basta disponibilizar o código-fonte, precisam ser indicadas quais versões das bibliotecas foram utilizadas e se as mesmas foram modificadas, incluir o código fonte delas. Todo o ferramental de compilação também deve ser mencionado. A praticidade de incluir uma imagem docker garante que os resultados possam ser reproduzidos rapidamente.

### 2.1. Avaliando propostas de voto com *blockchain* já existentes

Na literatura existem diversos trabalhos relacionados a voto e blockchain, em [Bistarelli et al. 2017] foi criado um sistema utilizando a *blockchain* pública do *bitcoin*. Primeiro há o problema do número de transações possíveis, o algoritmo do *bitcoin* foi criado de modo que a criaçã de novos blocos se faça a cada 10 minutos[Nakamoto 2009], cada bloco contém em média 2000 transações[7]. O número de votos possíveis por dia

[7] `https://outputs.today/` em 08/05/2019

seria:

$$24 * 6 * 2000 = 288.000$$

Levando em conta que as transações que não pagam taxas podem nunca ser inclusas e a taxa média [8] esta em 1,7 US$ este sistema não seria viável para grandes quantidades de votos. Dessa forma os *blockchains* públicos do *bitcoin* e do *ethereum*, não são dimensionáveis para votações de uma grande população. Seja pelo número de votos possíveis diários (*bitcoin* em 380 mil e *ethereum* 900 mil)[9] ou pelas taxas de cada transação (bitcoin 2,8 USD e ethereum 0,14 USD)[10]. Nesta implementação[11] em linguagem Go o autor não implementa verificação da autencidade das transações. No trabalho de [Hjalmarsson et al. 2018],[Cooley et al. 2018], [Shahzad and Crowcroft 2019] e [Wu and Yang 2018] são falado sobre os diversos aspectos de infraestrutura de uma eleição utilizando blockchain, mas não há implementação ou resultados de simulação. Em [Adiputra et al. 2018] e [Singh and Chatterjee 2018] são feitas propostas e parece ter havido uma implementação, mas não há resultados quantitativos. Os trabalhos de [Zhang et al. 2018], [Khoury et al. 2018]", [Yavuz et al. 2018], [M.C 2017], foram feitos com base no *blockchain* público do Ethereum, utilizando da plataforma pública e dos contratos nativos, não há resultados quantitativos. A escalabilidade da plataforma nas últimas semanas não ultrapassou 750 mil transações diárias[12]. Cada transação não pode indicar o voto de mais do que um eleitor, mesmo com os contratos inteligentes, isto torna inviável o uso para grandes populações.

### 2.2. Avaliando tecnologias de *blockchain* existentes

- BigChainDB[13]: um sistema que partiu de bancos de dados distribuídos e depois atribuiu características do blockchain.
- Chain Core[14] + Stellar: Uma plataforma de blockchain como serviço, o uso de sua infraestrutura é paga.
- Corda[15]: uma *blockchain* desenvolvida para uso corporativo, com uma versão aberta.
- Credits[16]: uma plataforma de desenvolvimento corporativa.
- Elements Blockchain Platform[17]: extensões para o protocolo do *bitcoin*.
- Eris:db[18]: extensões para o protocolo do *bitcoin*.
- Ethereum[19]: uma plataforma descentralizada que permite o uso de contratos dentro de sua própria *blockchain*.
- Quorum[20]: uma plataforma corporativa baseada no *Ethereum*.

[8] `https://bitinfocharts.com/comparison/bitcoin-transactionfees.html#3m` em 08/05/2019

[9] `https://bitinfocharts.com/comparison/transactions-btc-eth.html#3m`

[10] `https://bitinfocharts.com/comparison/transactions-btc-eth.html#3m`

[11] `https://github.com/codegoalie/votechain` em 08/05/2019

[12] `https://etherscan.io/chart/tx` em 05/2019

[13] https://www.bigchaindb.com

[14] https://chain.com

[15] https://www.r3.com/corda-enterprise

[16] https://credits.com

[17] https://elementsproject.org

[18] https://erisindustries.com

[19] https://www.ethereum.org

[20] https://www.goquorum.com

- Multichain[21]: uma extensão de código aberto criada a partir do código do *bitcoin*, projetada para transações financeiras de múltiplos ativos.
- Bitcoin[22]: Foi a criptomoeda inicial, criada em 2008, possui código livre e dezenas de desenvolvedores e milhões de usuários.
- Openchain[23]: uma plataforma corporativa para gerenciar ativos.
- HydraChain[24]: uma extensão do Ethereum para criação de blockchains com permissões.
- Hyperledger Fabric[25]: uma *blockchain* criada pela IBM, ela possui permissões e permite a execução de contratos inteligentes análogos ao *ethereum*.

Ainda existem plataformas de desenvolvimento criadas por grandes empresas que visam facilitar o desenvolvimento, entretanto as soluções ficam atreladas aquela empresa. IBM[26], Oracle[27], Microsoft[28], Amazon[29] possuem serviços semelhantes, cujo uso requer infraestrutura paga.

## 3. Desenvolvimento

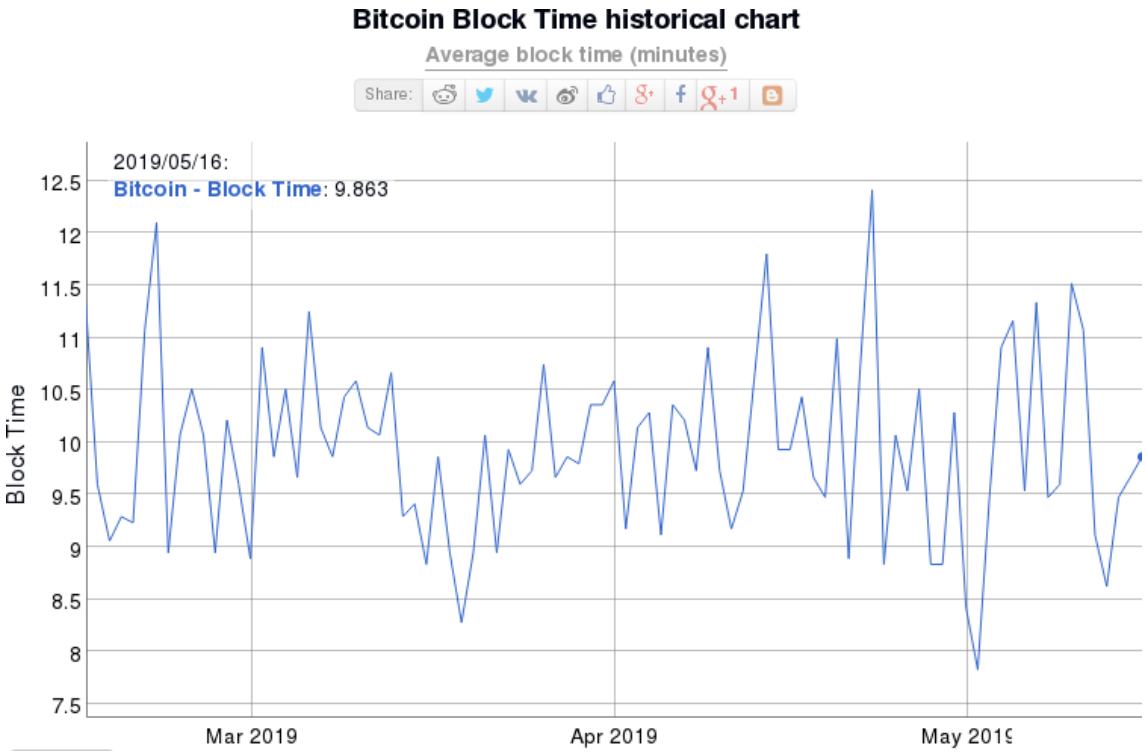
A escolha deste trabalho foi inicialmente criar um código do zero, porém isto foi feito apenas de forma didática para aprender os conceitos. Em seguida se baseou no código do *bitcoin*, este projeto foi o que iniciou a revolução do *blockchain*, desde sua criação não houve nenhuma falha de segurança grave, é a criptomoeda com maior capitalização (US$ 141 bilhões), valor individual (US$ 7 mil), maior valor mediano de transferências (US$ 509) e maior número de endereços ativos (772 mil). Essa importância do *bitcoin* significa que em qualquer momento existem dezenas de milhares de desenvolvedores e usuários procurando falhas em sua rede de modo a obter acesso ou criar *bitcoins* sem efetuar os cálculos necessários. O desafio do trabalho foi escalonar a operação, a rede do *bitcoin* foi projetada desde o princípio para gerar novos blocos a cada 10 minutos3, com cada bloco tendo em média 2000 transações, o algoritmo se adapta ao poder computacional da rede fazendo com que este intervalo se mantenha não importando o número de nodos da rede. Também conta com diversas funções, distribuição dos blocos, distribuição das transações, banco de dados otimizado para gravação e recuperação das informações, criação de uma rede independente, código criptográfico robusto. Também a cada novo desenvolvimento do código principal do *bitcoin*, as mudanças podem ser incorporadas retroativamente, não sendo necessária uma equipe de desenvolvimento própria e se utilizando do conhecimento da comunidade.

A ideia original com o código foi modificar a dificuldade mínima de cada bloco, permitindo uma geração mais rápida. Cada bloco gerado, ou minerado, recebe como prêmio pela participação na rede 50 *bitcoins*, cada moeda pode ser dividida em 100 milhões de unidades. Também modificando o intervalo e o tamanho dos blocos, permitindo uma geração mais rápida de blocos maiores. Os mineradores iriam distribuir frações

[21] https://www.multichain.com
[22] https://bitcoin.org
[23] https://www.openchain.org/
[24] https://github.com/HydraChain/hydrachain
[25] https://www.hyperledger.org/projects/fabric
[26] https://www.ibm.com/blockchain
[27] https://www.oracle.com/br/cloud/blockchain/
[28] https://azure.microsoft.com/en-us/services/blockchain-service/
[29] https://aws.amazon.com/pt/blockchain/

dos *bitcoins* para cada eleitor, processo o qual também gera transações precisando de um período para criação destes novos blocos. Qualquer nodo externo poderia se conectar a rede e incluir novos blocos, gerando para si *bitcoins* e podendo distribui-los, isso seria o equivalente a forja de votos. Isto poderia ser mitigado pela criação de uma rede de computadores protegida por *firewall*. O conceito do sistema de voto com *blockchain*, são servidores funcionando como nodos completos, que gerariam blocos e receberiam transações. No período de preparação estes servidores gerariam um par de chaves e um endereço para cada eleitor, fariam a criação das cédulas distribuindo os *bitcoins* nestes endereços. Essa associação eleitor e endereço seria de responsabilidade do governo, para distribuição das chaves uma autenticação é necessária, podendo ser uma senha cadastrada previamente ou até biometria. O problema desta ideia é que os recursos computacionais exigidos seriam cada vez maiores, também há o fato de que a maioria dos cálculos seria desperdiçado, resultando em tempo e energia gastos sem utilidade. O projeto final trabalhou modificando o tamanho dos blocos, o tempo de processamento entre cada bloco, a criação de transações especiais que criam *tokens* utilizando-os como cédulas, permissões de acesso, paralelização da apuração e da simulação de milhares de votos simultâneos. Estas modificações foram feitas com base no código do *bitcoin* e do *multichain*.

**Figura 3. Tempo de geração dos blocos no *blockchain* do *bitcoin*.**




Fonte: `https://bitinfocharts.com/comparison/bitcoin-confirmationtime.html`

### 3.1. Utilizando permissões de acesso

Um *blockchain* com permissões possui um controle de acesso, que dita quem pode administrar, conectar, enviar, receber, minerar. Algumas soluções já existem para isso como o Corda[Brown R. G. 2016], *Chain Core, Credits, HydraChain, BigchainDB*. Foi escolhida

a implementação do *multichain* por ser baseado e manter compatibilidade com o código do *bitcoin*, implementando este controle de acesso. Também foi escolhido pela licença do código ser aberta e qualquer trabalho derivativo também precisar ser aberto, isto concorda com os valores de pesquisa financiada com verba pública também ser aberta a todos.

O administrador possui as chaves iniciais, assim que se inicia uma blockchain, o software cria uma chave que assina os blocos e o identifica ao conectar com outros nodos. Esta chave concede permissões a qualquer outra chave. No sistema implementado as permissões padrões do são:

- Apenas nodos permitidos podem se conectar
- Apenas nodos permitidos podem criar blocos
- Todos os nodos podem enviar e receber transações
- Apenas transações permitidas podem ser enviadas
- Apenas nodos permitidos podem criar transações iniciais

Estas permissões são armazenadas dentro da própria *blockchain*, o nodo que tiver acesso e se conectar irá efetuar uma cópia de todos os blocos, transações e permissões.

### 3.2. Criando moedas não nativas dentro do blockchain

A criação de bens, tokens ou moedas não nativas dentro do *blockchain* é uma ideia que extende o conceito original, as transações não são mais das mesmas unidades, diferentes moedas podem ser trocadas. Um sistema de registro de uma casa de câmbio pode especificar que houve troca de dólares por euros entre dois usuários (identificados no *blockchain* pelas suas chaves). A cada eleição pode ser criada uma nova moeda e ser usada como cédula, essa cédula pode identificar o ano e local da eleição. Também permite que diferentes votações sejam realizadas ao mesmo tempo utilizando o mesmo *blockchain*.

Algumas[30] implementações[31] de blockchain oferecem a criação destes tokens. E entre elas o *multichain*, facilitando assim o reuso de código e a integração.

### 3.3. Segurança

Há vários níveis de segurança, onde cada um reforça o anterior. O primeiro deles é o acesso aos nodos através de uma requisição HTTP-RPC, esta possui uma senha, que pode ser divulgada aos clientes somente no dia desejado, isso evitaria que os nodos recebam requisições agentes externos. O segundo é a possibilidade de utilizar HTTPS, isto evitaria que os dados das transações sejam capturados, isto apenas divulgaria a informação de voto daquele cliente, caso a rede de entrada dos nodos seja comprometida isto divulgaria as intenções de votos de todos, os votos são assinados com as chaves privadas dos eleitores, mesmo que o voto seja capturado, ele não pode ser alterado, o https garantiria que o voto não possa ser lido, não teria a indicação do voto. O terceiro nível é a limitação das operações dos clientes com o nodo, que seriam apenas a autenticação e envio do voto, no código original do *bitcoin* podem ser requisitadas informações sobre transações, o tamanho da *blockchain*, performance do sistema, todos os blocos podem ser acessados por essa interface. A quarta medida de segurança é a geração das cédulas pelo administrador, somente ele possui a chave com permissão para tanto, não podem haver transações utilizando outras cédulas. O quinto nível é que os eleitores somente podem enviar para os

[30] https://github.com/OpenAssets
[31] https://www.openchain.org/

candidatos, as permissões de acesso são modificadas para que a lista de eleitores possam somente enviar e os candidatos somente receber, o voto dos próprios candidatos teria de ser registrado de outra forma. A parte mais delicada do sistema seria o banco de dados com as informações de usuários e suas respectivas chaves, caso haja vazamentos uma nova lista de chaves tem de ser gerada.

## 3.4. Possíveis problemas

### 3.4.1. Sybil Attack

No *bitcoin* quando uma entidade controla mais do que 50% da rede, ela pode decidir incluir blocos sem transações legítimas. No sistema proposto isto é evitado com o governo controlando a rede. E somente as transações criadas inicialmente podem ser transmitidas, somente o administrador tem autonomia pra criação das cédulas iniciais. Pela natureza aberta da apuração dos votos, qualquer manipulação feita pelo governo com os votos pode ser detectada, cada eleitor pode verificar se seu voto foi para o candidato correto e também a apuração dos votos.

### 3.4.2. Consenso Bizantino

Este problema é sobre o consenso entre as informações transmitidas, cada transação recebe uma identificação que pode ser usada para verifica-la. Também há o consenso entre os nodos sobre os blocos inclusos, todos terão a maior cadeia de blocos, pelo sistema proposto ser controlado, não há competição para geração de cadeias diferentes.

### 3.4.3. Transação duplicada

Um mesmo eleitor pode criar duas transações com saídas diferentes e a mesma entrada, mas somente a primeira sera validada recebendo um id da transação dentro da *blockchain*. Isto é um problema no *bitcoin* pelo fato de que há várias cadeias de blocos diferentes, no sistema proposto só há uma.

### 3.4.4. Criação de Votos Aleatórios (Interferência Externa)

A criação de um novo bloco não tem recompensa e somente as transações permitidas são inclusas. A quantidade de cédulas é fixa e criada antes da distribuição.

### 3.4.5. Inclusão de Blocos Aleatórios

No *bitcoin* um nodo pode entrar na rede e gerar um bloco com a dificuldade mínima correta, a recompensa por essa geração é um número de *bitcoins* que pode ser usado. Na implementação isto é evitado pois somente nodos autorizados podem fazer parte da rede.

#### 3.4.6. Brute-Force de Votos

Uma transação precisa de uma transação de origem e uma chave privada, esta chave privada pode tentar ser adivinhada, utilizando valores aleatórios. O custo computacional é altíssimo e a recompensa seria um único voto.

#### 3.4.7. Criptografia

A criptografia por trás das chaves *ECDSA* é robusta e a implementação usada também foi extensivamente testada pela indústria e academia. Caso ela seja quebrada, o código pode ser trocado, esta é uma das vantagens de se ter utilizado dos projetos *bitcoin* e *multichain*, uma falha grave dessas teria resposta imediata e por ser um trabalho derivado o sistema de votação pode incorporar as mudanças facilmente.

#### 3.4.8. Tamanho do blockchain

A base de dados com 32 milhões de votos, distribuídos e votados possui 28GB. Isto não é um problema para os servidores, chamados nodos, mas caso os eleitores precisassem ter um nodo para votar isto seria impraticável. Cada cliente, que pode ser um computador desktop, celular, página web, precisará apenas das informações das chaves privada e id da transação (sua cédula), estas informações são palavras-chaves de 58 caracteres e 36 respectivamente, a criação e assinatura da transação será local, não sendo necessário guardar os blocos.

### 3.5. Segurança

Há vários níveis de segurança, onde cada um reforça o anterior. O primeiro deles é o acesso aos nodos através de uma requisição HTTP-RPC, esta possui uma senha, que pode ser divulgada aos clientes somente no dia desejado, isso evitaria que os nodos recebam requisições agentes externos. O segundo é a possibilidade de utilizar HTTPS, isto evitaria que os dados das transações sejam capturados, isto apenas divulgaria a informação de voto daquele cliente, caso a rede de entrada dos nodos seja comprometida isto divulgaria as intenções de votos de todos. O terceiro nível é a limitação das operações dos clientes com o nodo, que seriam apenas a autenticação e envio do voto, no código original do *bitcoin* podem ser requisitadas informações sobre transações, o tamanho da *blockchain*, desempenho do sistema, todos os blocos podem ser acessados por essa interface. A quarta medida de segurança é a geração das cédulas pelo administrador, somente ele possui a chave com permissão para tanto, não podem haver transações utilizando outras cédulas. O quinto nível é que os eleitores somente podem enviar para os candidatos, as permissões de acesso são modificadas para que a lista de eleitores possam somente enviar e os candidatos somente receber, o voto dos próprios candidatos teria de ser registrado de outra forma. A parte mais delicada do sistema seria o banco de dados com as informações de usuários e suas respectivas chaves, caso haja vazamentos uma nova lista de chaves tem de ser gerada.

## 4. Resultados

Os testes foram feitos utilizando um servidor com:

- 32GB de memória RAM
- 2x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz (8 núcleos físicos, 16 virtuais)
- Discos rígidos mecânicos
- Rede 100Mbit

O código utilizado como base foi do projeto bitcoin e de outro código baseado no primeiro, chamado multichain. Foram criados clientes em modo texto em python, C++ e clientes gráficos em python. Nos testes de simulação de voto, foram criadas 500 instâncias nos computadores clientes, os quais possuíam:

- 8GB de memória RAM
- Processador Intel® Core™ i7-4790
- Discos rígidos mecânicos
- Rede 100Mbit

A segunda máquina utilizada como cliente possuía:

- 12GB de memória RAM
- 2x Intel(R) Xeon(R) CPU E5620 @ 2.40GHz (8 núcleos físicos, 16 virtuais)
- Discos rígidos mecânicos
- Rede 100Mbit

## 5. Distribuição

A distribuição dos votos consiste primeiramente da criação de uma transação especial com a quantidade total de votos a serem utilizados. Nenhum voto fora dessa transação pode ser computado pelos nodos. A partir de uma transação de origem, a próxima deve usar a quantidade exata de votos ou incluir uma saída extra com os votos restantes e um endereço. Há limitações no tamanho máximo do número de saídas da transação, nos testes o limite padrão de 4000 saídas foi utilizado. O processo de distribuição tem uma natureza obrigatoriamente sequencial, as primeiras 3999 cédulas são distribuídas entre os eleitores e a última saída para o endereço do administrador com o total de votos menos esses 3999. Essa transação gera um identificador que é usado na próxima iteração. Isso se repete até que todos os eleitores tenham recebido e o endereço do administrador ficaria com o resto das cédulas não utilizadas. Os resultados podem ser vistos na tabela 1. E a arquitetura na figura 4.

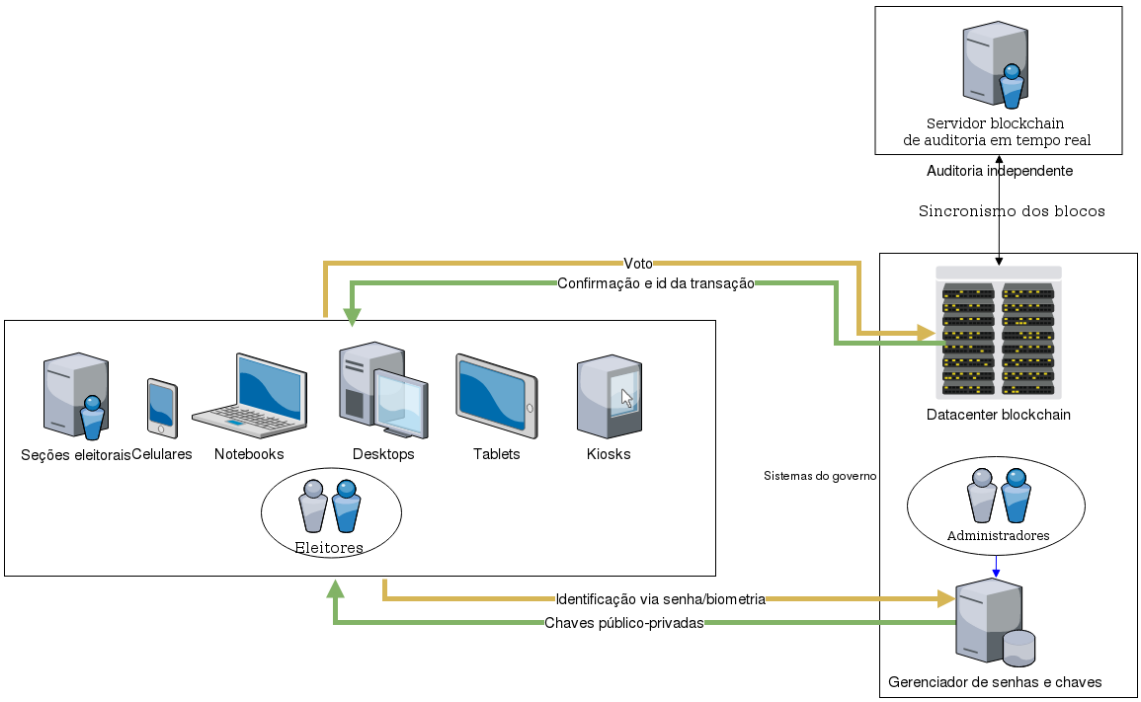
## 6. Simulando os votos

Os votos foram modelados como na figura 5. Utilizando de 500 clientes simultâneos, executando cada voto individualmente.

## 7. Apuração

O processo de apuração é um problema massivamente paralelo, são milhões de operações independentes umas das outras, para tanto foram criadas funções otimizadas para processadores de vários núcleos. As informações das transações contém individualmente um volume de dados muito pequeno, são milhões de operações de curtíssimo tempo. O processo é paralelizado utilizando *OpenMP*, onde cada *thread* recebe um bloco com várias transações, cada bloco pode conter um número variado de transações e cada transação tem um número variável de saídas, pra cada uma dessas saídas teve de ser feita a contabilidade. O escalonamento do trabalho e do número das *threads* é dinâmico por conta dessa variação.

**Figura 4. Arquitetura**




Fonte: Autor

**Tabela 1. Tempo para simulação da distribuição dos votos**

| Votos | Tempo | Transações |
|---|---|---|
| 1 milhão | 44,865s | 251 |
| 10 milhões | 509,859s | 2501 |
| 100 milhões | 7343,854s | 25007 |

**Tabela 2. Tamanho das bases de dados**

| Milhões de votos | Tamanho em MB |
|---|---|
| 1 | 630MB |
| 10 | 5400MB |
| 35 | 28000MB |
| 50 | 35000MB |
| 100 | 55000MB |

**Tabela 3. Tempo para apuração de 100 milhões de votos**

| Número de threads | Tempo de execução |
|---|---|
| 1 | 2929,786s |
| 2 | 1480,545s |
| 4 | 742,964s |
| 8 | 391,053s |
| 16 | 334,332s |
| 32 | 406,661s |

**Tabela 4. Tempo para apuração com diferentes populações**

| Número de votos | Tempo de execução |
|---|---|
| 10 milhões | 31,293s |
| 50 milhões | 176,852s |
| 100 milhões | 406,661s |

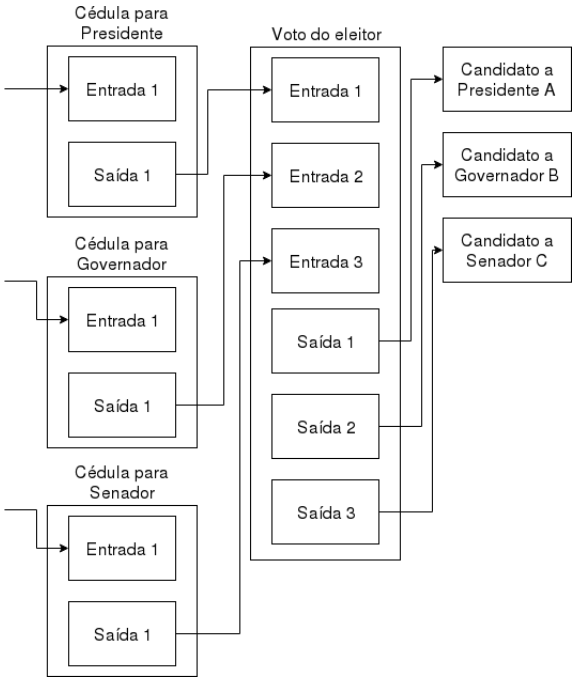
**Tabela 5. Tempo total da execução dos votos**

| Milhões de votos | Tempo de execução |
|---|---|
| 50 | 29h37m |

**Tabela 6. Tempo de execução de cada voto**

| Mínimo | Máximo | Mediana | Média |
|---|---|---|---|
| 0,709s | 35,231 | 0,799s | 1,064s |

**Figura 5. Modelagem de um voto como transação.**




**Fonte: Autor**

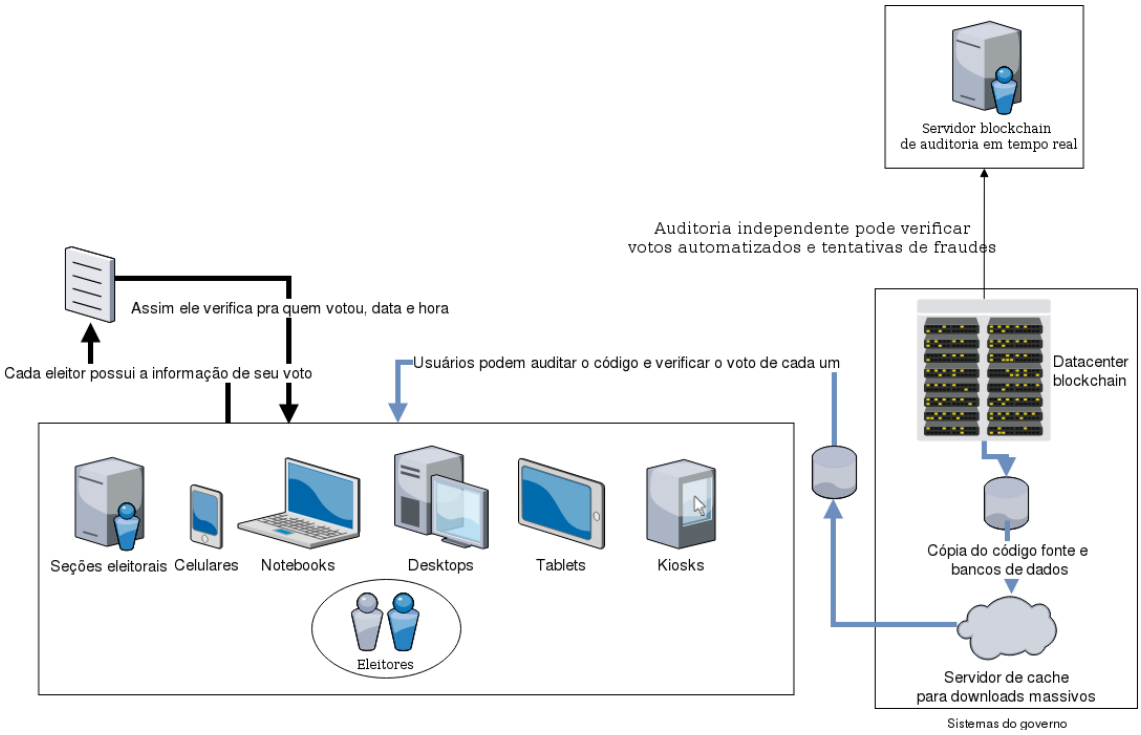
## 8. Auditoria

Segundo Rivest[Rivest 2006], um dos autores da chave de criptografia RSA e autor do Princípio da Independência de Software em Sistemas Eleitorais: "Um sistema eleitoral é independente do software se uma modificação ou erro não-detectado no seu software não pode causar uma modificação ou erro indetectável no resultado da apuração." De acordo com este princípio, o presente trabalho atende, pois sua implementação pode ser feita em outra linguagem ou arquitetura desde que respeitando as regras do *blockchain*, usando as mesmas funções matemáticas, regras do protocolo e com as mesmas regras de transação. A prova disso é que existem outros clientes bitcoin em outras linguagens, para a apuração dos votos, poderiam ser implementadas as regras nestas outras linguagens que interpretem as transações como votos.

O código fonte compactado fica abaixo de 10MB, isto inclui os fontes originais do *bitcoin* e *multichain* mais modificações e melhorias implementadas pelo autor. A imagem docker do ambiente de compilação com todas as bibliotecas instaladas ocupa 600MB instalado.

Cada voto ou transação, gera um identificador (listagem 8). Ele pode ser usado para recuperar os dados de voto e assim validar se a escolha do eleitor foi mantida ou manipulada. Esse identificador pode ser convertido para um *QR code* (listagem 7) para facilidade de uso. O eleitor interessado em auditar seu próprio voto pode fazer o download do código, a base de dados, compilar ele mesmo e validar as informações, nessa transação haverá a transação origem da cédula dele, pra quais endereços de candidatos ele enviou e a quantidade de votos.

**Figura 6. Processo de auditoria.**




Fonte: Autor

**Figura 7. QR Code verificador.**

Fonte: Autor

**Figura 8. Identificador da transação validada**

```
7d2572c5426faca62f37e6ab275fafd792869e9ee2f...
```

Fonte: Autor

## 8.1. Análise dos Resultados

Pelo monitoramento da implementação temos que inicialmente o número de transações por hora era de 2 milhões, mas ao término de 24 horas haviam sido executados 43 milhões de votos, aproximadamente 10% a menos em relação ao esperado inicialmente. Esta ineficiência pode ser devido a escrita das bases de dado em disco, a parte principal do programa responsável pela escrita em disco também acabou afetando o recebimento de novos votos, com o tempo isto foi acumulando. Também temos que o consumo de memória se manteve constante por volta de 1,5GB. O tempo para recebimento e consolidação das transações em blocos pode ser melhorado, utilizando um particionamento dos eleitores

em diferentes servidores, o resultado foi com um único servidor, a vantagem de se utilizar o código do *bitcoin* é que se já tem a implementação de código para redistribuir transações e blocos, efetivamente criando uma rede distribuída. Também pode-se criar várias *blockchains* de acordo com o tamanho da população, uma por estado, ou cidade, até mesmo bairro. A indicação de para qual nodo o eleitor seria direcionado ficaria logo após a identificação e envio da chave privada e *txid* da cédula. Na apuração o processo é feito em paralelo entre as várias blockchains regionais, nelas os resultados serão ainda mais rápidos pelo menor número de blocos em cada servidor. Em comparação com as 5 horas do sistema atual, o ganho foi considerável, onde um único servidor conseguiu apurar de forma correta 100 milhões de votos em pouco mais de 400 segundos. Os tempos para execução do código também são baixos, onde a mediana ficou em 0,8 segundos, isto possibilitaria a execução em dispositivos com pouco poder de processamento. A distribuição de 100 milhões de cédulas levou pouco mais de 2 horas, este processo seria no pior caso, em várias *blockchains* distribuídas em estados, cidades, bairros, o processo seria mais rápido ainda.

## 9. Conclusões

Os resultados foram animadores, levando em conta que a *blockchain* utilizada pelo *bitcoin* teve uma média diária de 380 mil transações e a *blockchain* do *ethereum* 900 mil[32], o sistema desenvolvido obteve 43 milhões utilizando a mesma estrutura de segurança do *bitcoin* com um processador lançado em 2010[33]. Os resultados serão melhores utilizando melhor hardware, mas também há limitações no algoritmo que precisam ser verificadas, a geração e inclusão de milhões de transações precisa evitar conflitos de hash e há várias travas para o acesso às estruturas de dados, para evitar condições de corridas nas *threads* concorrentes. O acesso a disco também requer milhares de operações não sequenciais por segundo, que poderiam ser beneficiadas utilizando de armazenamento flash. A busca por resultados quantitativos utilizando as proporções de uma eleição no Brasil, provaram que o *blockchain* pode ser uma alternativa viável e superior na questão de transparência, ainda que haja vários elementos que podem ser melhorados.

## Referências

Adiputra, C. K., Hjort, R., and Sato, H. (2018). A proposal of blockchain-based electronic voting system. In *2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 22–27.

Bistarelli, S., Mantilacci, M., Santancini, P., and Santini, F. (2017). An end-to-end voting-system based on bitcoin. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pages 1836–1841, New York, NY, USA. ACM.

Brown R. G., Carlyle J., G. I. H. M. (2016). Corda: An introduction.

Cooley, R., Wolf, S., and Borowczak, M. (2018). Blockchain-based election infrastructures. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–4.

[32] `https://bitinfocharts.com/comparison/transactions-btc-eth.html#3m`

[33] `https://ark.intel.com/content/www/us/en/ark/products/47925/intel-xeon-processor-e5620-12m-cache-2-40-ghz-5-86-gt-s-intel-qpi.html`

Hjalmarsson, F. P., Hreioarsson, G. K., Hamdaqa, M., and Hjalmtysson, G. (2018). Blockchain-based e-voting system. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 983–986.

Khoury, D., Kfoury, E. F., Kassem, A., and Harb, H. (2018). Decentralized voting platform based on ethereum blockchain. In *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, pages 1–6.

M.C, A. M. S. (2017). *Ethervoltz: um sistema de votação auditável baseado no blockchain ethereum*. Graduação em engenharia da computação, ETEP Faculdades/Faculdade de tecnologia de São José Dos Campos.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*.

Rivest, R.L; Wack, J. (2006). On the notion of "software independence" in voting systems.

Shahzad, B. and Crowcroft, J. (2019). Trustworthy electronic voting using adjusted blockchain technology. *IEEE Access*, 7:24477–24488.

Singh, A. and Chatterjee, K. (2018). Secevs : Secure electronic voting system using blockchain technology. In *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 863–867.

Wu, H. and Yang, C. (2018). A blockchain-based network security mechanism for voting systems. In *2018 1st International Cognitive Cities Conference (IC3)*, pages 227–230.

Yavuz, E., Koç, A. K., Çabuk, U. C., and Dalkılıç, G. (2018). Towards secure e-voting using ethereum blockchain. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–7.

Zhang, W., Yuan, Y., Hu, Y., Huang, S., Cao, S., Chopra, A., and Huang, S. (2018). A privacy-preserving voting protocol on blockchain. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 401–408.

# NOPL-Erlang: Programação multicore transparente em linguagem de alto nível

**Fabio Negrini[1], Adriano Francisco Ronszcka[1], Robson Ribeiro Linhares[1], João Alberto Fabro[1], Paulo Cézar Stadzisz[1], Jean Marcelo Simão[1]**

[1] Prog. de Pós-Graduação em Engenharia Elétrica e Informática Industrial Universidade Tecnológica Federal do Paraná (CPGEI-UTFPR)
80230-901 – Curitiba – PR – Brasil

{negrini,ronszcka}@alunos.utfpr.edu.br,
linhares,fabro,stadzisz,jeansimao}@utfpr.edu.br

***Abstract.*** *The growth of the computational capacity has been reached by means of the increment of cores inside a same processor. This solution, however, greatly increases the complexity of programming, forcing developers to make use of programming concurrency techniques. In this paper the NOPL is presented, a programing language for NOP with decoupling and non-sequential properties. NOPL is a high level multicore transparent programming language. This paper presents the NOPL language and its integration with Erlang architecture to explore concurrent execution. In the results it is possible to verify the expressive reduction in the execution time as the number of cores is increased.*

***Resumo.*** *O aumento da capacidade computacional dos processadores tem sido alcançado por meio do incremento de núcleos dentro de um mesmo processador. Esta solução, entretanto, aumenta consideravelmente a complexidade de programação, obrigando desenvolvedores fazerem uso técnica de programação concorrente. Neste artigo é apresentado a NOPL, linguagem própria do PON com propriedades desacoplante e não sequencial. NOPL é uma linguagem que permite a programação concorrente de maneira transparente e em alto nível. O artigo apresenta a linguagem NOPL e sua integração com a arquitetura Erlang para explorar execução concorrente. Nos resultados é possível verificar a redução significativa no tempo de execução à medida em que se aumenta o número de núcleos disponíveis.*

## 1. Introdução

O fato de os semicondutores estarem próximos do limite de velocidade tem instigado pesquisas ao redor do mundo para encontrar formas alternativas de aumentar o desempenho computacional [DeBenedictis 2017]. Neste sentido, o aumento da densidade de integração tem sido aproveitado pelos fabricantes para a implementação de múltiplos núcleos em uma mesma pastilha, o que é usualmente chamado de multicore. Embora, em tese, o aumento do número de unidades de processamento em paralelo permita aumentar o desempenho de execução da computação, na prática isto depende de software que explore adequadamente o paralelismo.

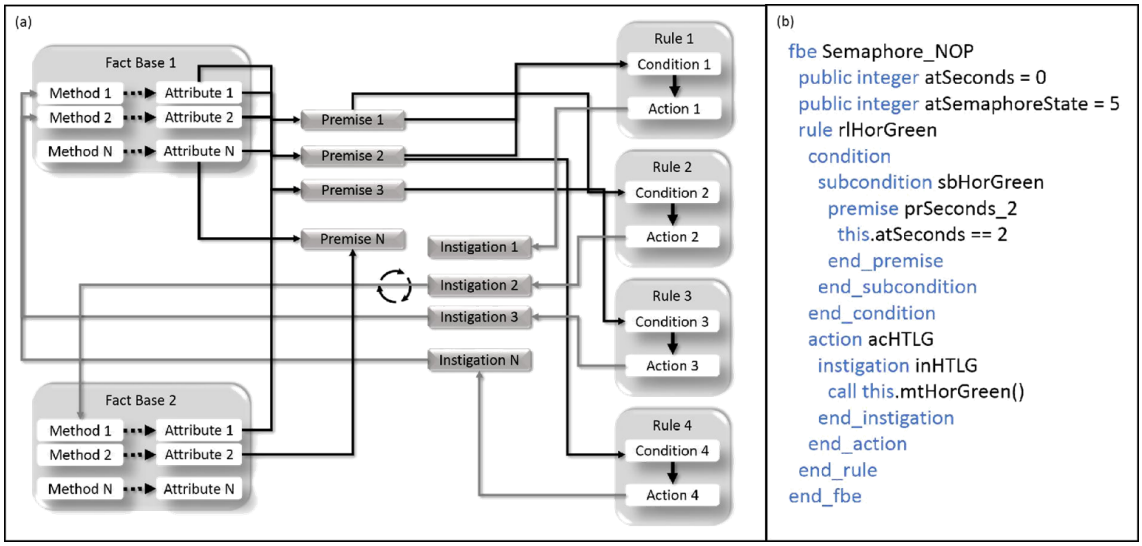



**Figura 1. a) Colaboração por notificações em PON [Ronszcka et al. 2017]**
b) Exemplo de código em NOPL

Entretanto, a construção de software de forma paralela e/ou adequada para execução em arquiteturas multicore impõe naturalmente dificuldades de abstração aos desenvolvedores em função justamente da nova dinâmica de execução paralela [Borkar and Chien 2011].

Neste contexto, este artigo explora uma abordagem alternativa de desenvolvimento de software: o Paradigma Orientado a Notificações (PON). O PON é, em suma, um paradigma emergente para o desenvolvimento de sistemas computacionais com maior nível de abstração, o que facilita a construção de programas que explorem adequadamente o paralelismo/distribuição em comparação com sistemas baseados em subparadigmas tradicionais, como Programação Procedimental, Programação Orientada a Objetos (POO) e Sistemas baseados em Regras (SBR) [Simão and Stadzisz 2009].

Neste artigo são apresentados avanços alcançados com a evolução NOPL (Notification Oriented Programming Language) [Ronszcka 2019] e seu compilador próprio, criando um novo target baseado na tecnologia Erlang. Este novo target de compilação é suportado por um framework sobre uma máquina virtual Erlang. A combinação da NOPL com este compilador específico proporciona a característica de programação em alto nível e execução concorrente de forma transparente.

## 2. Revisão da Literatura

### 2.1. PON e NOPL

O PON é um paradigma de programação emergente que apresenta similaridades a sistemas baseados em regras. Estruturalmente, entretanto, o software PON é representado na forma de entidades, nomeadamente as entidades chamadas elementos da Base de Fatos (FBE – Fact Base Element) e as Regras (Rules). As entidades FBE são utilizadas para representar objetos do mundo (real ou abstrato) em um sistema computacional, por meio de estados (atributos) e serviços (métodos). Os métodos PON, entretanto, são bastante pontuais, podendo apenas realizar operações matemáticas ou ações diretas, sem possuir lógica intrínseca de tomada de decisão (não possuem nem comandos de seleção (if ) nem estruturas de repetição (for/while). As entidades Rules, por sua vez, definem o cálculo lógico-causal a ser efetuado sobre os estados dos FBEs, controlando a execução de suas ações. A colaboração entre estes elementos ocorre por meio de notificações diretas das

entidades que constituem cada FBE (i.e., Attributes e Methods) e Rules (i.e., Premises, Conditions, Actions e Instigations) [Simão and Stadzisz 2009]. O diagrama apresentado na Figura 1a apresenta graficamente os relacionamentos entre as entidades citadas. Neste contexto é proposta a NOPL - a linguagem de programação baseada no paradigma PON que traz consigo os princípios fundamentais do PON: a) facilidade de desenvolvimento de software em alto nível; b) isenção de redundâncias estruturais, e c) desacoplamento explícito dos elementos. A tecnologia de compilação da NOPL consiste em um tradutor que se utiliza de um meta-modelo na forma de um grafo-framework permitindo sua derivação para uso em plataformas distintas [Ronszcka 2019]. A figura 1b apresenta um exemplo de código em NOPL.

### 2.2. Erlang

A linguagem Erlang surgiu na década de 1980 nos laboratórios de Ciência da Computação da Ericsson para suportar aplicações distribuídas e tolerantes a falhas. Em vez de fornecer processos que compartilham memória, cada processo Erlang é executado em seu próprio espaço de memória e possui seu próprio heap e pilha evitando interferências indevidas entre os processos. Erlang aplica a abordagem de concorrência Modelo de Ator de forma que os processos se comunicam entre si via passagem de mensagens assíncronas [Cesarini and Thomson 2009]. Contudo, apesar das inúmeras facilidades, a granularidade dos atores ainda depende da habilidade do desenvolvedor em pensar e construir módulos suficientemente desacoplados.

## 3. Materiais e métodos

### 3.1. PON e o modelo de atores

O modelo de atores de Erlang [Hewitt et al. 1973] requer algum meio para prover atores com granularidade apropriada para melhor aproveitar o balanceamento de processamento dos atores em multicore. Neste sentido, as tecnologias PON e Erlang poderiam ser conjugadas. A proposta deste artigo é o aproveitamento da arquitetura concorrente Erlang com aderência dos elementos do paradigma PON em processos com estados que se comunicam por meio de mensagens. Após uma análise minuciosa de cada elemento PON, foi possível chegar a uma modelagem de alto nível de cada um de seus elementos em forma de microatores. Com esta modelagem, cada elemento apresentado na Figura 1a é traduzido em um processo concorrente Erlang e o mecanismo de notificações, por sua vez, traduzido em mensagens assíncronas.

### 3.2. Framework NOP Erlang/Elixir

Uma vez feita a análise de aderência dos elementos do NOP para elementos com comportamento de atores, avançou-se a para a codificação. Optou-se pela utilização da linguagem Elixir, pois esta é totalmente imersa na plataforma Erlang e ainda disponibiliza mecanismos essenciais para a programação estruturada [Thomas 2018]. Estes mecanismos resultantes podem ser utilizados de maneira independente e estão disponíveis para a comunidade Elixir para testes e avaliações. O resultado desta codificação é apresentado em forma de pacote HEX para facilitar sua distribuição e utilização. Em suma, criou-se um framework NOP Erlang/Elixir que opera sobre a lógica do PON e não sobre a tradicional lógica funcional-imperativa sobre a qual normalmente os sistemas são programados nesta plataforma.

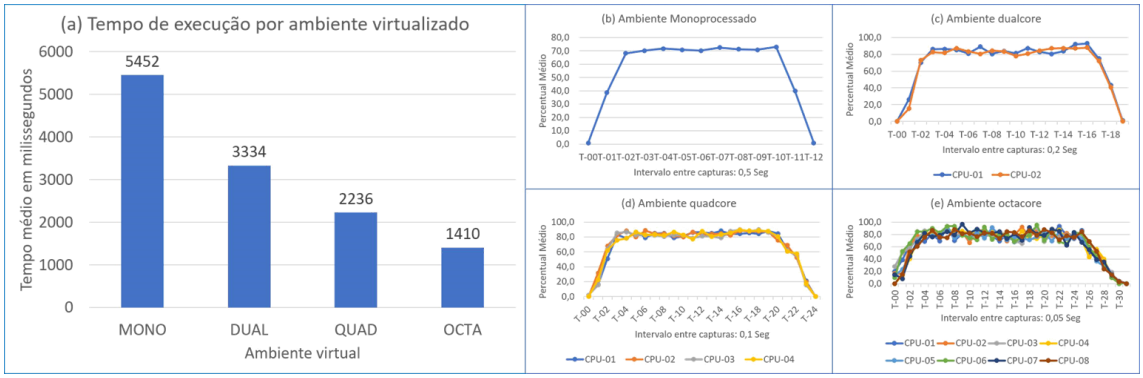



**Figura 2. Resultados das simulações em ambientes EC2 tipo T2 Amazon AWS**

### 3.3. Target Framework NOP Erlang/Elixir

Por fim, seguiu-se a etapa de construção do compilador NOPL especializado para o target framework NOP Erlang/Elixir. Esta etapa é baseada no método MCPON (Método de Compilação PON) que permite a construção de compiladores por meio de um grafo de entidades PON e um conjunto de diretrizes que facilitam a integração da linguagem NOPL com o novo target gerado [Ronszcka 2019]. Isto posto, os elementos identificados no grafo-framework são então convertidos em elementos do framework e suas ligações devidamente alinhadas. O resultado é uma solução que permite desenvolver aplicações em NOPL e automaticamente compilá-las para microatores em PON sobre a plataforma Erlang. Assim, o desenvolvedor dispõe de recursos para um melhor aproveitamento da capacidade de concorrência através da arquitetura concorrente Erlang.

## 4. Resultados e discussões

### 4.1. Experimento CTA

Para avaliar a capacidade de processamento concorrente multicore com a solução proposta, nomeada de tecnologia NOPL-Erlang, propôs-se como experimento um programa em NOPL para controle de trânsito automatizado (CTA) definido em [Renaux et al. 2015]. O objetivo do CTA é alternar os estados de semáforos durante um ciclo de noventa segundos. Este ambiente foi reproduzido para um conjunto de dez quadras horizontais por dez quadras verticais totalizando cem semáforos. Depois, foram simulados ciclos de 2000 segundos em sequência para todos os semáforos.

Foi utilizado um conjunto de quatro ambientes virtualizados do tipo T2 Ubuntu Server 14.04 LTS 64 bits e armazenamento SSD disponibilizados pela Amazon. Os resultados podem ser vistos na Figura 2. A Figura 2a representa o tempo médio de execução (em milissegundos) em cada ambiente virtualizado. Como é possível perceber no cenário apresentado, há uma redução de 73,27% tempo de execução do programa de simulação quando comparado com o ambiente mono core com o octa core. As Figuras 2b, 2c, 2d e 2e demonstram uma distribuição balanceada entre os núcleos respectivamente para ambientes mono, dual, quad e octa core, sugerindo que a execução aproveitou o potencial disponível dos núcleos de forma balanceada em prol de uma otimização e redução do tempo total de execução.

## 5. Conclusão e trabalhos futuros

Como é possível verificar, com a tecnologia NOPL-Erlang, conforme o mesmo programa é executado em ambiente com múltiplos núcleos (multicore), o seu tempo de execução se reduz, ao passo que os núcleos permanecem todos com balanceamento de utilização durante todo o processamento, demonstrando que a execução está conseguindo se ocupar de todos os núcleos para este experimento, que avaliou ambientes de até oito núcleos. Todo este aproveitamento é alcançado com programação em alto nível e estruturada, sem interferência ou conhecimento do programador em ambientes multicore devido ao trabalho de união da linguagem NOPL e o desacoplamento implícito próprio do PON com a arquitetura concorrente Erlang, por meio de um framework que explora adequadamente as características do PON. Somente com Erlang, tal concorrência seria possível com o desenvolvimento de alguma técnica de programação paralela demandando conhecimento adicional. Como trabalhos futuros sugere-se avaliações mais profundas em relação à utilização com maior quantidade de núcleos para melhor avaliar o comportamento desta nova tecnologia. Outro trabalho futuro sugerido é o de aprimorar a tecnologia NOPL-Erlang para garantir determinismo conforme detalhado em [Simão and Stadzisz 2010], os quais não foram explorados nos experimentos apresentados neste artigo.

## Referências

Borkar, S. and Chien, A. A. (2011). The future of microprocessors.

Cesarini, F. and Thomson, S. K. (2009). Erlang Programing. O'Reilly Media.

DeBenedictis, E. P. (2017). It's time to redefine moore's law again. Computer, 50(2):72–75.

Hewitt, C., Bishop, P., and Steiger, R. (1973). A universal modular actor formalism for artificial intelligence.

Renaux, D. P. B., Linhares, R. R., Simão, J. M., and Stadzisz, P. C. (2015). Cta conops. http://www.dainf.ct.utfpr.edu.br/~douglas/CTA_CONOPS.pdf.

Ronszcka, A. (2019). Método para a criação de linguagens de programação e compiladores para o paradigma orientado a notificações em plataformas distintas. Teste. Doutorado em Computação Aplicada - CPGEI. Universidade Tecnológica Federal do Paraná (UTFPR) Curitiba - PR.

Ronszcka, A., Valença, G., Linhares, R., Stadzisz, P., and Simão, J. (2017). Notification-oriented paradigm framework 2.0: An implementation based on design patterns. IEEE Latin America Transactions, 15(11).

Simão, J. M. and Stadzisz, P. C. (2009). Inference based on notifications: A holonic metamodel applied to control issues. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 39(1):238–250.

Simão, J. M. and Stadzisz, P. C. (2010). Mecanismo de resolução de conflito e garantia de determinismo para o paradigma orientado a notificações (pon). Pedido de patente nro PI1000296-0. Data de depósito no INPI: 02/2010.

Thomas, D. (2018). Programing Elixir. The Pragmatic Bookshelf.

# Avaliando a Detecção e o Tratamento de *Flash Crowds* Utilizando o SimGrid

**Vitor Pinheiro David**[1], **Caio César A. Sampaio**[1], **Eduardo dos Santos Costa**[1], **Ubiratam de Paula** [1]

[1] Universidade Federal Rural do Rio de Janeiro (UFRRJ)

vitor.pito@gmail.com, caio.cesarsampaio@hotmail.com

eduardo00100@hotmail.com, upaula@ufrrj.br

***Abstract.*** *This article proposes a simulation, where cloud resources can be dynamically hired in order to mitigate the effects of flash crowds events. This simulation was developed through the integration of SimGrid simulator with existing solutions in the literature for the flash crowds detection and handling problems. The initial tests were satisfactory and promising. For future works, we intend to perform more tests with larger scenarios.*

***Resumo.*** *Este artigo propõe uma simulação, onde recursos da nuvem podem ser contratados dinamicamente com a finalidade de mitigar os efeitos dos eventos de flash crowds. Esta simulação foi desenvolvida através da integração do simulador SimGrid com soluções já existentes na literatura para a detecção e tratamento de flash crowds. Os testes iniciais foram satisfatórios e promissores. Futuramente, pretende-se realizar mais testes com cenários maiores.*

## 1. Introdução

Com o aumento da popularidade da Internet e, consequentemente, da quantidade de seus usuários, conteúdos podem se tornar virais muito rapidamente. Nestes casos, os conteúdos podem ficar indisponíveis ou com baixa qualidade de serviço (QoS). Estes eventos são denominados *flash crowd* e precisam ser detectados e tratados de maneira rápida e eficiente.

Existem muitas definições de *flash crowd* na literatura [Stavrou et al. 2002, Wendell and Freedman 2011, Atajanov et al. 2007]. Em [Stavrou et al. 2002], uma *flash crowd* é definida como um fenômeno resultante de um aumento repentino e imprevisível na popularidade de um conteúdo online. Em [Wendell and Freedman 2011], os autores definem *flash crowd* como um período no qual as taxas de requisições para um domínio crescem exponencialmente. Dessa forma, pode-se concluir que um evento de *flash crowd* acontece quando as taxas de acesso de um conteúdo ou domínio sofrem um aumento além do esperado para o tráfego normal. Por exemplo, um site de notícias possui um taxa média de acessos por dia. Entretanto, quando algum fato noticiado possui grande interesse ou popularidade, como ataques terroristas ou desastres naturais, as taxas de acesso sofrem um grande aumento, ultrapassando a taxa média normal esperada. Também é possível observar um cenário similar nos períodos de inscrições de disciplinas nos sites/sistemas das universidades, onde o fluxo de requisições aumenta consideravelmente.

Várias soluções para este problema já foram estudadas por diversos pesquisadores [Wendell and Freedman 2011, Sladescu et al. 2013, Atajanov et al. 2007,

de Paula et al. 2015, de Paula 2015]. Em [de Paula 2015], foi apresentado um mecanismo de detecção de *flash crowd*, chamado FCD (*Flash Crowd Detection*), que utiliza conceitos de Teoria da Informação, como entropia e correlação. A maior vantagem do FCD em relação à literatura relacionada é que não é necessário definir uma escala de crescimento para o número de acesso, ou seja, não é preciso tentar quantificar o aumento do número de acessos através de determinadas características, como crescimento exponencial. Além disso, é possível calcular os valores máximos para as entropias e correlação total utilizando apenas o número de conteúdos, que é conhecido.

Também em [de Paula 2015], o tratamento deste eventos foi modelado matematicamente como um problema de programação inteira, chamado *Flash Crowd Handling Problem - Integer Programming* (FCHP-IP). Neste modelo, foi incluído a possibilidade de contratação de recursos elásticos (Computação em Nuvem) para atender de forma satisfatória as demandas de requisições. Como o tempo para resolver as formulações matemáticas pode ser elevado (dependendo do tamanho dos dados de entrada), uma heurística, chamada FCHP-ILS, também foi proposta. A FCHP-ILS é baseada na metaheurística *Iterated Local Search* (ILS)[Talbi 2009, Lourenço et al. 2003] e por quatro componentes: (i) um procedimento para gerar uma solução inicial viável atribuindo cada requisição ao servidor com menor custo e com largura de banda disponível; (ii) uma busca local que tenta encontrar uma nova solução viável de menor custo a partir da solução atual; (iii) uma estratégia de perturbação que visa escapar de ótimos locais ainda distantes dos ótimos globais; e (iv) um critério de aceitação elitista para novas soluções, onde apenas soluções com menor custo são aceitas e utilizadas nas iterações seguintes. Por fim, ainda em [de Paula 2015], o mecanismo de detecção FCD e a heurística FCHP-ILS foram avaliados teoricamente e em pequenos cenários executados na nuvem da Amazon EC2 [Amazon Web Services 2020].

Devido ao alto custo monetário para avaliar estas soluções em ambientes reais, neste artigo é apresentada uma proposta de simulação para este problema utilizando o simulador SimGrid [Casanova et al. 2014]. Desta forma, pode-se avaliar a qualidade das soluções em cenários maiores e que representam situações reais de *flash crowds*.

O simulador SimGrid foi escolhido, mais especificadamente seu módulo *s4u*, pelas seguintes razões: (i) ser amplamente utilizado pela comunidade científica, fato comprovado pela grande quantidade de artigos publicados que o utilizam[1], (ii) ter atualizações recentes, atualmente, na versão $3.24$[2], e (iii) ter fórum ativo de colaboradores[3]. Não foram encontrados trabalhos na literatura relacionada que utilizam simuladores de ambiente de nuvens de computadores para o problema abordado.

## 2. Simulação Realizada

Para avaliar as soluções propostas em [de Paula 2015], foi necessário construir uma simulação com a estrutura representada na Figura 1. Esta estrutura é baseada nos experimentos reais que foram realizados na nuvem da Amazon EC2. A diferença é que todos os servidores e usuários são implementados no simulador.

Na Figura 1, é possível observar que as requisições dos clientes são recebidas

[1] `https://simgrid.org/usages.html`
[2] `https://github.com/simgrid/simgrid`
[3] `https://lists.gforge.inria.fr/pipermail/simgrid-user/`

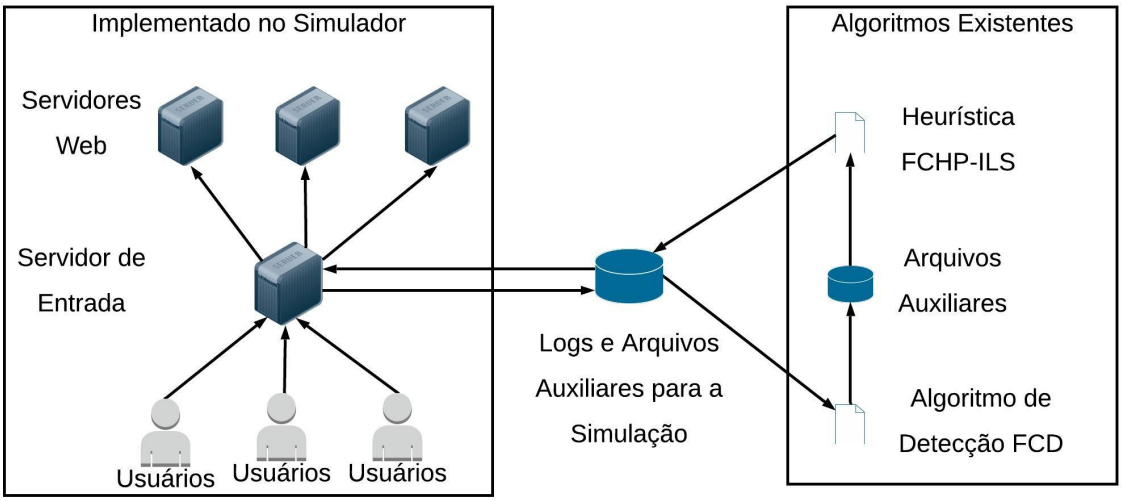



**Figura 1. Proposta de simulação com o SimGrid para o problema.**

por um servidor de entrada. Este, além de registrar as requisições no arquivo de *log*, também as redireciona para um servidor Web capaz de atendê-las. O mecanismo FCD é continuamente executado tendo como entrada o arquivo de *log*. Caso um início ou término de *flash crowd* seja detectado, a heurística FCHP-ILS é executada para verificar se a solução atual precisa ser modificada. Essa modificação pode um reposicionamento dos conteúdos nos servidores já existentes, uma contratação de novos servidores ou ambos. Este retorno para o simulador é feito através de leitura e escrita em arquivos.

A utilização de arquivos é uma solução mais elegante, porém a troca de dados no módulo *s4u* do SimGrid funciona através de *mailboxs*. As *mailboxs* realizam a comunicação entre atores que tenham uma rota de comunicação definida entre si. Os atores são criados a partir de *hosts*. O *host* é a estrutura de dados que define as características dos atores derivados dela. Além disso, essa estrutura é utilizada na criação das rotas de comunicação entre atores e define o que será representado, por exemplo, um servidor ou um usuário.

Atores são instâncias de *hosts*, ou seja, são as entidades que realmente executam durante a simulação. Suas características, como poder de processamento, número de núcleos, entre outros, são definidas pelo seu *host*. Cada ator tem uma função associada a ele e sempre que um ator é criado, essa função começa a executar e quando ela acaba o ator é excluído.

A simulação é executada a partir da criação de atores e da troca de mensagens entre eles. A criação dos atores é possível por código ou pela leitura de arquivos XML de entrada. As Figuras 2 e 3 ilustram exemplos destes arquivos. Na Figura 2, são definidos dois *hosts*, dois *links* e uma rota (composta pelos dois *links*). Na Figura 3, é definida qual função será executada pelo ator associado a um *host*. As informações necessárias para a criação destes arquivos são adquiridas de *traces* reais de *flash crowds*, que também foram utilizados em [de Paula 2015].

A contratação de recursos na nuvem e novas replicações são realizadas no simula-

```
<?xml version'1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">

<platform version="4.1">
  <zone id="first zone" routing="Full"><!-- Inicia uma rede -->
    <host id="C1" speed="1Mf"/> <!-- Inicia um host, nesse caso um cliente -->

    <host id="S1" speed="1Mf"/> <!-- Nesse caso um servidor -->

    <link id="C1" bandwidth="100Bps" latency="0ms"/> <!-- Inicia um link -->

    <link id="S1" bandwidth="1000MBps" latency="0ms"/>

    <route src="C1" dst="S1"> <!-- Inicia uma rota entre a origem e o destino -->
      <link_ctn id="C1"/> <!-- Referencia ao link já existente com id = C1 -->
      <link_ctn id="S1"/>
    </route>
  </zone>
</platform>
```

**Figura 2. Exemplo de XML utilizado pelo SimGrid para definir a rede.**

```
<?xml version'1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">

<platform version="4.1">

  <actor host="C1" function="request"> <!-- Inicia um ator com host = C1
  e esse ator executa a função request -->
    <argument value="exemplo"/> <!-- valor que é passado como parametro
    para a função do ator-->
  </actor>

</platform>
```

**Figura 3. Exemplo de XML utilizado pelo SimGrid para iniciar os atores.**

dor através das informações contidas em um arquivo auxiliar, gerado a partir do resultado da heurística FCHP-ILS. Os recursos elásticos são representados como servidores já definidos no arquivo XML, porém os atores que os representam só são criados de acordo com o arquivo auxiliar.

As simulações foram realizadas para reproduzir os dois cenários reais apresentados em [de Paula 2015], com *traces* de 1041 e 1798 requisições de um dia completo. Os resultados foram compatíveis e obtidos em menos de um segundo de simulação.

## 3. Conclusão e Trabalhos Futuros

Neste artigo foi apresentada uma proposta de simulação utilizando o SimGrid integrado às soluções para a detecção e tratamento de *flash crowds* propostas em [de Paula 2015]. Os resultados iniciais são promissores e compatíveis com os resultados encontrados nas execuções reais na nuvem da Amazon, considerando os mesmos *traces*. O desempenho da proposta também se mostrou satisfatório, apresentando tempo de simulação inferior a um segundo nos testes realizados. Como trabalhos futuros, espera-se realizar mais testes, principalmente em cenários reais e com diferentes tamanhos de eventos de *flash crowds*.

## 4. Agradecimentos

Os autores gostariam de agradecer à FAPERJ pela bolsa de iniciação científica concedida.

## Referências

Amazon Web Services, I. (2020). Amazon EC2. `https://aws.amazon.com/pt/ec2/`.

Atajanov, M., Shimokawa, T., and Yoshida, N. (2007). Autonomic multi-server distribution in flash crowds alleviation network. In *Emerging Directions in Embedded and Ubiquitous Computing*, volume 4809 of *Lecture Notes in Computer Science*, pages 309–320. Springer Berlin Heidelberg.

Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014). Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917.

de Paula, U. (2015). *Detecção e Tratamento de Eventos de Flash Crowd em Nuvens Computacionais*. PhD thesis, Universidade Federal Fluminense - Instituto de Computação.

de Paula, U., Drummond, L., de Oliveira, D., Frota, Y., and Barbosa, V. C. (2015). Handling flash-crowd events to improve the performance of web applications. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 769–774, New York, NY, USA. ACM.

Lourenço, H., Martin, O., and Stützle, T. (2003). Iterated local search. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 320–353. Springer US.

Sladescu, M., Fekete, A., Lee, K., and Liu, A. (2013). A polymorphic model for event associated workload bursts. In *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on*, pages 119–125.

Stavrou, A., Rubenstein, D., and Sahu, S. (2002). A lightweight, robust p2p system to handle flash crowds. *SIGCOMM Comput. Commun. Rev.*, 32(3):17–17.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Wiley Publishing.

Wendell, P. and Freedman, M. J. (2011). Going viral: flash crowds in an open CDN. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, IMC '11, pages 549–558, New York, NY, USA. ACM.


--