# Config Hell: Entendendo o impacto do aumento de arquivos de configuração em projetos open-source

**Paulo Henrique Horta Borges**[1], **Hudson Silva Borges**[1]

[1]Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)
Caixa Postal 79.070-900 – Campo Grande – MS – Brasil

{paulo.horta,hudson.borges}@ufms.br

***Abstract.*** *Modern software development is highly dependent on tools and third-party dependencies. Pursuing the goal of easing integration and adaptation of such solutions to a wide range of clients, it's very common for those tools to offer custom configurations to its users. Related to the increase in the quantity of tools needed to develop modern applications, many projects face something called Configuration Hell. In this project, it's proposed an early investigation on the subject of the evolution of configuration files in open-source software repositories. Furthermore, it is proposed an early analysis into the correlation between the evolution in the quantity of configuration files and software metrics reported in the literature. Preliminary results show that, in fact, there was a sizeable increase in the number of configuration files over the years. Additionally, a correlation between the increase in the number of configuration files and some software metrics was identified, such as cognitive complexity. This project's results show the relevance of new studies for further understanding the real impact and consequences of this growth.*

***Resumo.*** *O desenvolvimento de software moderno é altamente dependente de ferramentas e dependências externas. Visando facilitar a integração e adaptação de tais soluções para uma ampla gama de clientes, é muito comum que tais ferramentas ofereçam formas de configuração personalizadas a seus usuários. Associada ao crescimento na quantidade de ferramentas necessárias para desenvolvimento de aplicações modernas, atualmente diversos projetos enfrentam o que é chamado de Configuration Hell. Neste trabalho, é proposta uma investigação inicial sobre a evolução de arquivos de configuração em repositórios de software open-source. Além disso, é proposta uma análise inicial sobre a correlação entre a evolução da quantidade de arquivos de configuração e métricas de software adotadas pela indústria. Resultados preliminares mostram que, de fato, houve um crescimento muito grande na quantidade de arquivos de configuração ao longo dos anos. Além disso, foram identificadas correlações entre o crescimento no número de arquivos e algumas métricas de qualidade, como complexidade cognitiva. Os resultados deste trabalho demonstram a importância de novos estudos para entendimento do real impacto e consequências deste crescimento.*

## 1. Introdução

Com a democratização da informática, a atividade de desenvolvimento de software torna-se ubíqua, presente em todos os aspectos da vida cotidiana moderna, desde o desenvol-

vimento de aplicações de comunicação até portais web que hospedam os mais variados conteúdos.

Dessa forma, é de suma importância para empresas o estudo de procedimentos para melhorar a qualidade do software produzido, diminuir custos de desenvolvimento e manutenção do software e diminuir a quantidade e frequência de problemas encontrados por usuários.

Um ponto de discórdia para o desenvolvimento moderno de software é a inserção de códigos e ferramentas de terceiros no projeto, levando à adição de uma quantidade elevada de arquivos de configuração. Essa problemática é discutida pela comunidade, questionando sobre o impacto dessa inserção na qualidade do software e no seu desenvolvimento.

Com isso, este trabalho teve como objetivo realizar uma investigação inicial sobre o histórico e evolução desses arquivos de configuração nos projetos e verificar a correlação de tais alterações com métricas de qualidade de software tradicionais. Assim, foi realizada um estudo de caso de um projeto moderno e altamente popular para desenvolvimento de aplicações web, o `mui/material-ui`. Os resultados obtidos mostram um aumento altamente significativo no número de arquivos de configuração ao longo de dez anos, e a análise de correlação com métricas de software mostrou uma alta correlação com métricas indicativas de problemas de desenvolvimento, como, por exemplo, bugs, complexidade cognitiva e violações.

O restante deste trabalho está organizado como a seguir. Na Seção 2 são abordados tópicos essenciais para o entendimento deste trabalho. Na Seção 3 são apresentados e discutidos trabalhos relacionados ao tópico estudado. Na Seção 4 são detalhados os procedimentos usados neste estudo e na Seção 5 são apresentados os resultados encontrados. Por fim, uma discussão sobre os resultados encontrados é apresentada na Seção 6 e as conclusões do trabalho são feitas na Seção 7.

## 2. Background

O trabalho apresentado neste artigo tem como um de seus objetivos a compreensão do impacto do crescimento no número de arquivos de configuração em projetos de software e seu impacto na qualidade mensurável dos mesmos. Assim, esta seção fornece informações básicas necessárias sobre duas perspectivas: (i) configuração de software e (ii) métricas de qualidade de software. Na Seção 2.1, são apresentadas evidências do crescimento do problema mencionado em comunidades de software atuais e na Seção 2.2, é abordado o conceito de métricas de software, que têm sido usadas como forma de avaliar a qualidade de sistemas ao longo do tempo.

### 2.1. Configuração de Software e Config Hell

Em projetos de software modernos, é muito comum a existência de arquivos de configuração [Siegmund et al. 2020]. De fato, configuração de software tem sido um tópico bastante explorado na literatura. Por exemplo, [Siegmund et al. 2020] argumenta que o termo "configuração" tem sido usado de diferentes formas, sendo que arquivos de configuração são a forma preferida por desenvolvedores para configuração de software. Tal formato tem sido frequentemente utilizado como forma de facilitar o processo de

personalização e adaptação de dependências sem a necessidade de construir diferentes versões de um mesmo software.

Arquivos de configuração podem ser encontrados em diversos domínios, como, por exemplo, bibliotecas de componentes gráficos (e.g., para customização de estilização de componentes web), configuração de ambiente e compilação (e.g., informações para transpilação de projetos TypeScript), controle de versionamento (e.g., com informações do servidor remoto), verificação e aplicação de estilos e boas práticas (e.g., bibliotecas como *eslint* e *prettier* para JavaScript, que verificam e aplicam estilos pré-definidos pelos desenvolvedores), entre outros.

A presença desses arquivos é comumente discutida no cenário de desenvolvimento de software. Contudo, nos últimos anos, o excesso de arquivos de configuração, mesmo em projetos mais simples, começou a gerar discussões e questionamentos em comunidades de desenvolvedores. Por exemplo, um artigo de opinião no blog do projeto *open-source* DENO, plataforma de execução para a linguagem JavaScript/TypeScript e um dos projetos mais populares na plataforma do GitHub, o autor descreve a situação atual dos arquivos de configuração em projetos de desenvolvimento web como "Config Hell".

Este termo, que pode ser traduzido como "Inferno dos Arquivos de Configuração", é caracterizado pela quantidade extrema de arquivos de configuração apresentados em um projeto recém-criado da linguagem [Jiang 2023]. No artigo mencionado, o autor apresenta a problemática da onipresença de arquivos de configuração em projetos criados na linguagem JavaScript/TypeScript. Para isso, o mesmo usa como exemplo um projeto da linguagem *JavaScript* com mais de 30 arquivos de configuração, contando com configurações de estilização, de controle de versionamento, de monitoramento de erros e de arquivos de configuração da própria linguagem. Ao final, o autor argumenta que esse cenário torna o processo de desenvolvimento mais complexo e menos produtivo, já que desenvolvedores devem estar atentos a questões não diretamente envolvidas ao objetivo do negócio.

## 2.2. Métricas de Software

É de suma importância para o processo de desenvolvimento de software a pesquisa de fatores que podem afetar a qualidade de um projeto e a produtividade de desenvolvedores. Ao longo do tempo, estudos e ferramentas foram propostos com intuito de fornecer formas de avaliação da qualidade e produtividade por meio de métricas extraídas de artefatos e processos de software.

Métricas de software são medidas responsáveis por demonstrar de forma concreta indicadores de qualidade, atribuindo valores para o processo por meio de regras pré-estabelecidas com o intuito de capturar dados, permitindo que sejam comparados com outros momentos ou outros processos similares conforme definido por [Norman Fenton 2015].

Dessa forma, foram desenvolvidos softwares com o intuito de medir a qualidade de um software baseado em valores pré-determinados, utilizando-se de métricas com o intuito de mensurar automaticamente "Boas práticas de programação", que são padrões conhecidos de desenvolvimento, comumente associados com boa manutenibilidade, compreensão, ausência de erros e vulnerabilidades. Uma dessas ferramentas é a

análise estática de código, que se baseia na busca automatizada de erros, vulnerabilidades e situações que possam dificultar a leitura e manutenção em uma base de código, aumentando sua complexidade.

Dentre as métricas medidas, podem ser encontradas métricas que medem a dificuldade de realizar manutenção do código, como possíveis erros. Alguns exemplos podem ser destacados como: *bugs* (número de defeitos encontrados no código fonte), *code smells* (trechos de código que não necessariamente apresentam defeitos, mas sim situações que dificultam a leitura e compreensão, como no exemplo citado por [Kaur and Singh 2023] de funções muito extensas) e a complexidade ciclomática (mede a complexidade de compreensão de um código, mensurado pelo número de instruções executadas em um trecho de código conforme proposto por [Misra 2006]).

Com a problemática apresentada, foi pesquisada a existência de trabalhos que faziam uma analise sobre a evolução e crescimento de arquivos de configuração em projetos de software e sobre seu possível impacto na qualidade do software porém, não foram encontrados trabalhos que investigassem esses pontos.

## 3. Revisão de Literatura

Em um estudo publicado por [Rishita Mullapudi 2021], os autores argumentam que é necessário um desenvolvimento rápido com poucos problemas para se manter competitivo em tempos modernos e que a predição de qualidade pode ajudar a manter esse ciclo. Para isso, realizaram um estudo de caso com o projeto de software Eclipse, analisando fatores de desenvolvimento como mudanças de última hora em arquivos (incluindo arquivos de configuração), propondo uma árvore de decisão que consiga predizer a qualidade do software com base em ações dos desenvolvedores. Embora os autores considerem os arquivos de configuração, não é feita uma análise específica sobre tais artefatos, sua evolução e seu impacto sobre o projeto.

Em outro estudo publicado por [Raab 2015], o autor argumenta que arquivos de configuração são essenciais para o desenvolvimento de software moderno e que uma grande parte dos erros são causados por erros nestes artefatos, sendo sintomas do estado da arte de desenvolvimento de software moderno. O trabalho propõe uma especificação de como a configuração deve ser feita, propondo um banco de dados de chave-valor para armazenar as configurações realizadas. Este trabalho se aprofunda na proposição de uma metodologia para evitar os problemas decorrentes do número extensivo de arquivos de configuração e seus impactos, mas não investiga a evolução desses arquivos em projetos e foca no gerenciamento de arquivos e solução de problemas causados por arquivos de configuração.

Na dissertação de [Lydall 2018], é argumentado que a customização de software gerenciada por arquivos de configuração tem efeitos negativos na qualidade estrutural e funcional do software. Para isso, o autor realiza um estudo de caso com três aplicações diferentes, verificando o nível de customização por linhas de código e comparando com métricas de manutenibilidade encontradas. Seus resultados mostram uma baixa correlação entre a manutenibilidade do projeto e o aumento de defeitos. Apesar do autor correlacionar o número de regras de configuração com métricas de software, não pondera sobre a quantidade de arquivos diferentes e não realiza uma análise histórica sobre o aumento de arquivos de configuração.

Segundo o estudo de [Rezvan Mahdavi-Hezaveh 2021], softwares com configurações para ligar/desligar funcionalidades, descrevendo os tipos de funcionalidade desativáveis, mostram quais os possíveis problemas dessa prática. Em seguida, realiza um estudo de caso analisando os artefatos gerados por essa prática, obtendo como resultado a quantidade de empresas que utilizam sistemas de gerenciamento de configurações. No entanto, o estudo apresenta os impactos negativos de uma vertente dos arquivos de configuração, não abordando arquivos de configuração como um todo, além de não realizar um estudo de correlação da qualidade do software com a presença dessa prática.

## 4. Metodologia

Com intuito de entender melhor o crescimento desordenado de arquivos de configuração em projetos *open-source* e o impacto na qualidade interna destas aplicações, este estudo foi conduzido seguindo 3 passos. Inicialmente, deu-se inicio à seleção do projeto para o estudo (Seção 4.1), e, em seguida, foram definidas heurísticas para a coleta de dados com o intuito de obter informações sobre arquivos de configuração no projeto (Seção 4.2) e, por fim, a extração de métricas de qualidade de software, com o proposito de realizar a análise estatística das métricas pelo tempo (Seção 4.3). Cada um dos passos é apresentados a seguir.

### 4.1. Seleção do projeto para estudo

Para a seleção do repositório de estudo, este estudo optou inicialmente pela busca de projetos relevantes na plataforma GitHub. O GitHub é atualmente a plataforma de codificação social mais popular para projetos *open-source* e oferece ferramentas abertas que permitem a coleta de dados por pesquisadores, portanto, oferecendo boas oportunidades para este estudo.

Além disso, considerando que este estudo busca entender um fenômeno recente e ainda pouco explorado, optou-se pela condução do estudo com foco em um único projeto com intuito de entender melhor os desafios e oportunidades da área.

Considerando também que grande parte dos relatos do problema de *configuration hell* surgiram de comunidades de desenvolvedores da linguagem de programação JavaScript, e outras derivadas, este estudo optou pela seleção de um projeto relevante para a comunidade JavaScript. De fato, a linguagem JavaScript tem sido reportada na literatura como uma das mais populares, se destacando pela quantidade de projetos entre os mais populares da plataforma GitHub, além também de liderar por vários anos seguintes o *ranking* de popularidade e uso segundo a própria plataforma.[1]

A seleção do projeto iniciou-se com a identificação dos top-25 repositórios mais populares na plataforma GitHub. Esse processo foi conduzido diretamente pela interface web da plataforma por meio da opção de busca avançada. Basicamente, restringiu-se a busca por repositórios com no mínimo uma estrela e escritos na linguagem JavaScript ou TypeScript ordenados pela quantidade de estrelas do projeto. O uso de estrelas como métrica de popularidade foi inspirada por trabalhos na literatura, principalmente o trabalho de Borges e Valente (2018)[Borges and Valente 2018].

[1] `https://github.blog/2023-11-08-the-state-of-open-source-and-ai/`, acessada em 20/06/2024.

Após a análise dos top-25 repositórios, optou-se pela condução das atividades sobre o projeto Material UI (`mui/material-ui`).[2] Dentre os motivos que levaram a optar por este repositório, destaca-se que o projeto é um *framework* alto nível de componentes web com uma grande e ativa comunidade de desenvolvedores, com quase 93K estrelas e 470 *releases*.

### 4.2. Definição de heurística e coleta de dados

Arquivos de configuração são tipicamente caracterizados por serem informações estruturadas que podem ser escritas em arquivos de diferente formatos. Embora atualmente alguns formatos sejam mais populares, como JSON (JavaScript Object Notation) e YAML (Yet Another Markup Language), tais arquivos são frequentemente encontrados em diferentes formatos e não existe um padrão de uso claro seguido por bibliotecas e desenvolvedores de software. Este aspecto do problema abordado é considerado como um dos grandes desafios desta pesquisa considerando a necessidade de uma abordagem automatizada que possa ser replicada futuramente à diversos outros projetos.

Neste trabalho, a heurística adotada foi composta pela inclusão de arquivos com extensões conhecidamente relacionadas associadas a arquivos de configuração, como `.json`, `.yaml`, `.ini`, etc., associado à localização do mesmo na estrutura do projeto. A seguir são apresentados detalhes da heuristica proposta:

1. Tamanho do arquivo: Critério é satisfeito se o arquivo possui menos de 10KB de tamanho (tamanho suficientemente pequeno escolhido pela experiência empírica de tamanho de arquivos de configuração), por exemplo “config.yml”;
2. Nome do arquivo: Padrões do nomenclatura normalmente são impostos à arquivos de configuração, contendo usualmente nomes como “config”, “settings”, arquivos iniciados com ”.”(denotando arquivos usualmente escondidos pelo gerenciador de arquivos do sistema), como por exemplo “config.yml”;
3. Localização do arquivo: Arquivos de configuração são mais comumente encontrados no diretório raiz do projeto e excluindo arquivos de configuração de dependências externas, como por exemplo “.browserslistrc”;
4. Formato do arquivo: Arquivos de configuração para projetos JavaScript usualmente são encontrados em diversos formatos, sendo eles dos tipos “json”, “toml”, “yaml”, “yml”, “xml”, “properties”, “lock” e arquivos sem extensão, como por exemplo “.gitignore”. Também é realizada a exclusão de arquivos por extensão, sendo escolhidos com base nos formatos encontrados no projeto que não são classificados como arquivos de configuração como “svg”, “md”, “txt”, como `README` (Leia-me) encontrado no projeto (que apesar de cumprir os critérios 1, 2 e 3, não se trata de um arquivo de configuração pela extensão)

Regras especificas de inclusão e exclusão também foram definidas por constituem convenções ou padrões conhecidos, sendo elas:

1. Arquivos “toml” são automaticamente considerados arquivos de configuração, por se tratar de um formato cujo intuito é a formatação de arquivos de configuração, exemplo: “netlify.toml”;

[2] `https://github.com/mui/material-ui`, acessado em 20/06/2024

2. Arquivos sem extensão no diretório raiz são automaticamente considerados arquivos de configuração, visto que esses arquivos comumente se tratam da configuração de ferramentas de terceiro para gerenciamento do projeto (como configuração de linter ou de Git), exemplo: ".eslintignore";
3. Arquivos com sufixo de nome "rc" são automaticamente considerados arquivos de configuração, por se tratarem de uma nomenclatura conhecida por ser utilizada para arquivos de configuração, como, por exemplo ".babelrc";
4. Arquivos dentro de diretório (ou sub-diretórios) chamados "node_modules" são automaticamente ignorados, porque mesmo que possam ser arquivos de configuração, são arquivos de configuração de dependências, não do próprio projeto.

Com os critérios propostos, são selecionados arquivos que atinjam conjuntos de critérios (e que não foram automaticamente incluídos ou excluídos pelos critérios específicos), filtrando arquivos que cumpram pelo menos dois desses critérios que indicariam arquivos de configuração, visto que atingir apenas um desses critérios causa a seleção de arquivos falso-positivos, como por exemplo um arquivo chamado "config.js", que passa no critério de nomenclatura, mas falha no critério de extensão. Dessa forma, foram selecionados os seguintes conjuntos:

- Conjunto de arquivos que possuam nomes relacionados à configuração (regra 2) e extensões que remetem a arquivos de configuração (regra 4) e/ou possuam menos de 10KB de tamanho (regra 1), exemplo: "tsconfig.json", que inclui a palavra config e a extensão json;
- Conjunto de arquivos que possuam menos de 10KB (regra 1) ou possuam nome que atinja a regra 2 ou estão na lista de extensões comuns de configuração (regra 4) e estão no diretório raiz do projeto (regra 3), exemplo: ".eslintignore";

Por fim, a identificação e coleta de informações dos arquivos de configuração existentes para o projeto `mui/material-ui` foi feita usando um *script* automatizado que consulta a API REST, da própria plataforma GitHub, por arquivos com extensões que atendem às heurísticas definidas. Para cada arquivo identificado, este *script* também extrai informações adicionais, como o caminho do arquivo no projeto, o tamanho do arquivo e a data que o mesmo foi adicionado ao repositório. Como a API não permite adicionar critérios de exclusão à busca, os critérios de exclusão foram aplicados após a coleta.

### 4.3. Extração de métricas

Para a extração de métricas, optou-se pelo uso da ferramenta SonarQube na versão *community*. SonarQube[3] é uma ferramenta de análise estática de código amplamente utilizada na indústria de desenvolvimento de software e por pesquisadores para análise de projetos. Além disso, a ferramenta permite a análise de projetos escritos em diversas linguagens de programação, incluindo *TypeScript*, a linguagem do projeto MUI/MATERIAL-UI. Neste trabalho, foram consideradas as seguintes métricas:

- *Blocker Violations*: Problema que age como um impedimento para a correta execução da aplicação ou desenvolvimento de novas funcionalidades;

[3] `https://www.sonarsource.com/products/sonarqube/`, acessada em 20/06/2024.

- *Bugs*: Erros no código que podem causar comportamentos inesperados/indesejados;
- *Code Smells*: Métrica correspondente a práticas de programação que podem apresentar um aumento na dificuldade de manutenção;
- Complexidade Cognitiva: Métrica que mede a legibilidade e compreensibilidade do código. É considerado o uso repetido e consecutivo de estruturas condicionais, de operadores lógicos e de sequências que quebram o fluxo de código conforme definido por [Campbell 2023];
- Linhas de código: Métrica que mensura o número de linhas que contenham pelo menos um caractere que não seja espaço em branco, tabulação ou parte de comentários, também conhecida pelo termo "ncloc";
- Vulnerabilidades: Trechos de código que possuem vulnerabilidades exploráveis por terceiros ou credenciais importantes mantidas no projeto (*e.g.*, um token de acesso exposto);

Objetivando identificar a evolução dos valores das métricas descritas ao longo do tempo, decidiu-se por extrair os valores destas métricas para todas as marcações de versões presentes no repositório do projeto. Em especial, foram consideradas todas as marcações do tipo *tag* registradas no sistema de versionamento do projeto. A decisão pela análise de tais registros se dá pela relevância destes para o projeto, pois, em geral, implicam em lançamento de novas versões que agregam contribuições relevantes ao longo do tempo. Ao todo foram identificadas 277 tags no repositório MUI/MATERIAL-UI, sendo a primeira a de título `v0.2.0`, publicada em 02/12/2014, e o último a de título `v5.15.20`, publicada em 21/05/2024. A recuperação do estado do repositório em cada registro de *tag* foi realizada utilizando-se do comando `git checkout`.

Por fim, para a extração das métricas mencionadas em cada um dos pontos do sistema, i.e., tags, foi desenvolvido um *script* na linguagem *Python* para a coleta das métricas a partir da API HTTP do SonarQube. Essencialmente, foi implantada uma instância da ferramenta localmente e, de forma iterativa, foram computados e coletados os resultados da análise em cada uma das versões do projeto MUI/MATERIAL-UI marcadas pelas suas respectivas *tags*.

## 5. Resultados

Com objetivo de verificar se, realmente, arquivos de configuração tem crescido em quantidade ao longo do tempo, este estudo começou verificando o crescimento destes artefatos no repositório MUI/MATERIAL-UI. A Figura 1 apresenta a evolução na quantidade de arquivos de configuração, identificados pelos critérios apresentados na Seção 4. É possível observar que a quantidade de arquivos no repositório analisado evoluiu consideravelmente, passando de 1 na versão `v0.2.0`, publicada em 2014, 263 na versão mais recente, o que representa um crescimento de 26200% ao longo de dez anos.

Ademais, o maior aumento no número de arquivos ocorreu na versão `v5.14.8`, registrada em agosto de 2023, com a adição de 75 arquivos de configuração. A maioria desses arquivos adicionados neste período são originados da criação de novos projetos de exemplo de uso da biblioteca com diferentes tecnologias (por exemplo, o uso com a *framework* de desenvolvimento web NextJS). De fato, o repositório analisado demonstrou uma atividade importante e frequentemente encontrada em projetos de software *open-source*: a disponibilização de exemplos de uso de suas soluções. Nesses cenários, os

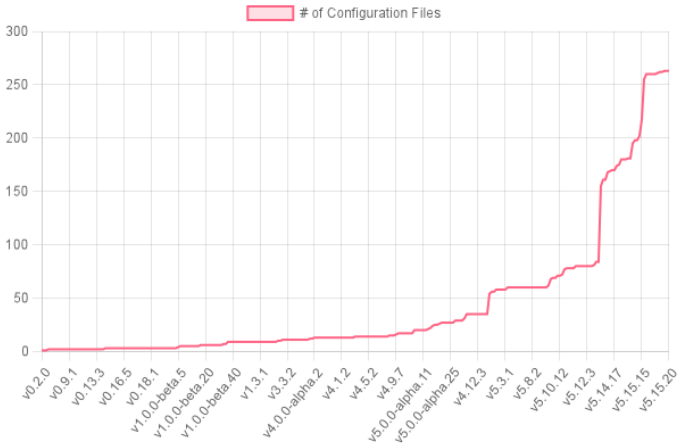


**Figura 1. Evolução no número de arquivos de configuração**

mantenedores do projeto enfrentam o desafio de, além de manter arquivos necessários para a correta execução do próprio sistema, também fazer a manutenção de arquivos de configuração considerados não essenciais, como os dos próprios exemplos fornecidos. Da mesma forma, foi observado um considerável aumento de 46 arquivos de configuração na versão `v5.15.12` de março de 2024, que corresponde ao lançamento do pacote `Pigment CSS` no projeto, acompanhado de exemplos de uso da tecnologia.

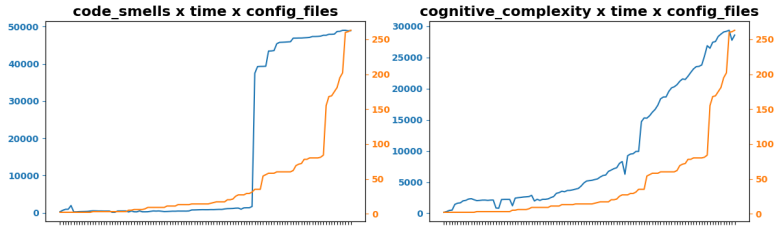


**Figura 2. Evolução de Code Smells e Complexidade Cognitiva *vs* Arquivos de Configuração**

Além de analisar a evolução nos arquivos de configuração, este estudo também analisou e comparou a evolução dos arquivos de configuração em relação à métricas de software, reportadas na Seção 4.3. A Figura 2 apresenta as evoluções do número de *code smells* e da complexidade cognitiva medida ao longo do tempo, comparando com o crescimento no número de arquivos de configuração. Em relação à code smells, observa-se o crescimento muito grande a partir da versão `v5.0.0` do projeto. Contudo, não foi observado uma similaridade no crescimento de ambas as métricas. De fato, ao aplicar o teste de correlação de Spearman foi identificado uma alta probabilidade correlação (valor-p menor que 0,05, rejeitando a hipótese nula, de que as amostras não são correlacionadas com 95% de confiança) porém, com moderada intensidade de correlação (coeficiente de 0,86). Por outro lado, ao observar a evolução da complexidade cognitiva do projeto, os resultados sugeriram que ambos poderiam estarem relacionadas. Os resultados obtidos a partir da aplicação do teste de Spearman mostraram que a métrica está fortemente relacionada ao número de arquivos de configuração, com 95% de confiança e um coeficiente de correlação de Spearman 0,948.

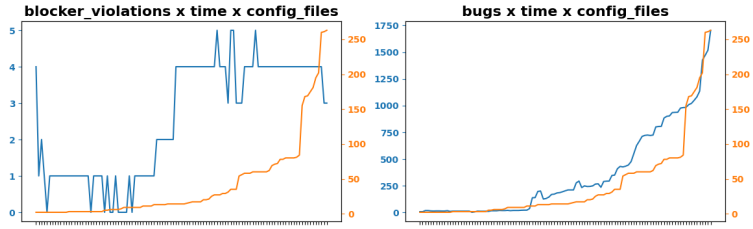


**Figura 3. Evolução de Blocker Violations e Bugs *vs* Arquivos de Configuração**

A Figura 3 apresenta a evolução no número de *blocker violations* e de *bugs* ao longo do tempo em relação ao crescimento de arquivos de configuração. Em relação à blocker violations, os resultados do teste de Spearman mostram apenar uma fraca correlação (amostras correlacionadas com 95% de confiança e coeficiente de correlação de Spearman de 0,55), porém, enquanto o mesmo teste aplicado à evolução de *bugs* mostra uma forte correlação com 95% de confiança e coeficiente de correlação de 0,94, simbolizando um alto nível de correlação entre a métrica e os arquivos.

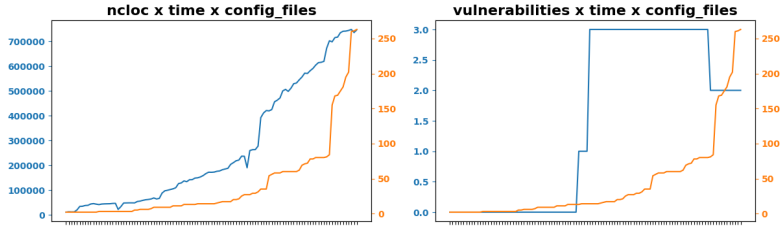


**Figura 4. Evolução de Linhas de Código e Vulnerabilidades *vs* Arquivos de Configuração**

Por fim, a Figura 4 apresenta a evolução no número de linhas de código (*ncloc*) e número de vulnerabilidades, comparado ao número de arquivos de configuração. É possível observar uma alta correlação entre o número de linhas de código com o número de arquivos de configuração, que é corroborado pela aplicação do teste de Spearman, demonstrando correlação com 95% de confiança e um coeficiente de correlação de Spearman de 0,96.Ao aplicar o mesmo processo para o número de vulnerabilidades, da mesma forma é possível demonstrar correlação, porém com coeficiente de correlação de Spearman menor, de 0,77.

## 6. Discussão

Com a investigação realizada no projeto MUI/MATERIAL-UI, foi possível observar um crescimento significativo de arquivos de configuração utilizados em projetos, apresentando um aumento de 1 arquivo na primeira versão registrada (`v0.2.0`) para 263 arquivos na versão mais recente e analisada neste trabalho. Assim, os resultados deste estudo apresentam evidências iniciais que reforçam o sentimento compartilhando entre diversos autores sobre o aumento de arquivos de configuração e reportados neste trabalho na Seção 2.1.

Considerando a relação entre arquivos de configuração e métricas de software, os resultados deste trabalho mostraram uma baixa intensidade de correlação entre arquivos

de configuração e métricas como blocker violations e vulnerabilidades. Por outro lado, os resultados também mostram uma correlação significativamente alta com complexidade cognitiva, *bugs* e número de linhas de código. Embora não seja possível apontar uma relação de causalidade, podemos destacar as seguintes hipóteses:

- *Complexidade cognitiva*: À medida que projetos evoluem, os mesmos tendem a se tornarem mais complexos e difíceis de se manter. Uma das hipóteses que podem ajudar a explicar tal relação está na necessidade de integrar novas ferramentas e bibliotecas para apoiarem novas funcionalidades. Embora este estudo foque no crescimento do número de arquivos de configuração, é necessário que, futuramente, também seja analisado se a complexidade de manutenção de tais arquivos também aumentam à medida que a complexidade do sistema aumenta.
- *Bugs*: Bugs são considerados potenciais problemas que podem impactar negativamente o sistema. Neste estudo, identificou-se também a correlação com o crescimento no número de arquivos de configuração. Estudos da literatura apontaram que problemas de configuração de software podem representar ameaças à funcionalidade do sistema, contudo, é necessário entender quantos e quais problemas estão diretamente relacionados ao crescimento no número de arquivos de configuração.
- *Linhas de Código:* Assim como complexidade cognitiva, evoluir um sistema geralmente implica no crescimento no número de artefatos no sistema, como linhas de código. Neste sentido, arquivos de configuração estão diretamente relacionados, uma vez que ferramentas de medição consideram vários arquivos de configuração como parte dos artefatos mensuráveis. Artigos promovidos por desenvolvedores e críticos do crescimento da quantidade de arquivos de configuração apontam a necessidade de simplificar esses recursos, o que pode contribuir para evitar o crescimento desordenado da complexidade e arquivos em um sistema.

É necessário destacar que este trabalho teve como objetivo dar um primeiro passo em direção ao entendimento sobre o fenômeno no crescimento do número de arquivos de configuração, denominado de "Config Hell" por desenvolvedores e comunidades de software. Neste sentido, foram conduzidos estudos quantitativos sobre a evolução de tais valores no repositório MUI/MATERIAL-UI. Cabe ressaltar que correlação não implica em causalidade, ou seja, não é possível dizer, a partir dos resultados reportados, que um fenômeno causa o outro.

Por fim, os resultados apresentados neste trabalho podem não representar o comportamento geral do fenômeno, uma vez que existem diversos outros fatores que podem influenciar os comportamentos reportados. Por exemplo, aplicações de outros domínios e linguagens de programação podem apresentar outros comportamentos. Neste estudo, buscou-se conduzir um estudo exploratório inicial a partir de um projeto *open-source* importante para o desenvolvimento de software moderno e desenvolvido na linguagem JavaScript/TypeScript, que é a comunidade com mais relatos sobre o problema. Além disso, dada a inexistência de critérios claros para identificação de arquivos de configuração, este estudo ofereceu uma proposta inicial para apoiar trabalhos futuros.

## 7. Conclusão

Nos últimos anos, comunidades e desenvolvedores de software tem observado e reportado um crescimento na quantidade de arquivos de configuração necessários para o desenvolvi-

mento de aplicações, mesmo que simples. Esse fenômeno foi denominado como "Config Hell". O problema apontado refere-se à dificuldade em manter tais sistemas além dos riscos de introdução de problemas por erros na configuração dos mesmos.

Com objetivo de prover um entendimento inicial sobre o problema apontado, este estudo propôs a investigação do crescimento no número de arquivos de configuração, e a relação com outras métricas de software, no repositório `mui/material-ui`, o framework mais popular de componentes web para construção de páginas dinâmicas usando da biblioteca React.

Para isso foi feito uma análise e definição de critérios de inclusão e exclusão para detecção de arquivos de configuração no projeto, além de coletar e analisar a evolução do crescimento destes arquivos ao longo das principais marcações (*tags*) de versões do projeto.

Ao comparar o crescimento do número de arquivos com métricas tradicionais de software, obtidas por meio da ferramenta Sonarqube, os resultados mostraram uma correlação significativa com três métricas: complexidade cognitiva, bugs e número de linhas de código.

Embora não seja possível apontar uma relação de causalidade, os resultados evidenciam preocupações externadas por desenvolvedores e comunidades nos últimos anos. Assim, é necessário que trabalhos futuros aprofundem a análise por meio da inclusão de mais projetos à análise, assim como busquem entender melhor a influência dos arquivos de configuração nas métricas analisadas e outras não consideradas neste estudo.

## Referências


[Borges and Valente 2018] Borges, H. and Valente, M. T. (2018). What's in a GitHub star? understanding repository starring practices in a social coding platform. *Journal of Systems and Software*, 146:112–129.

[Campbell 2023] Campbell, G. A. (2023). {*Cognitive Complexity*} *a new way of measuringunderstandability*.

[Jiang 2023] Jiang, A. (2023). Node.js's config hell problem. Technical report, Deno Land Inc.

[Kaur and Singh 2023] Kaur, S. and Singh, S. (2023). Object oriented metrics based empirical model for predicting "code smells" in open source software. *Journal of The Institution of Engineers (India): Series B*, 104.

[Lydall 2018] Lydall, G. (2018). Quality impact of configuration and customisation on configurable software.

[Misra 2006] Misra, S. (2006). A complexity measure based on cognitive weights. *International Journal of Theoretical and Applied Computer Sciences Volume Number*, 1:1–10.

[Norman Fenton 2015] Norman Fenton, J. B. (2015). *Software Metrics A Rigorous and Practical Approach*. Taylor & Francis Group, 3rd edition.

[Raab 2015] Raab, M. (2015). Safe management of software configuration. In *CAiSE 2015 Doctoral Consortium*.

[Rezvan Mahdavi-Hezaveh 2021] Rezvan Mahdavi-Hezaveh, Jacob Dremann, L. W. (2021). Software development with feature toggles: practices used by practitioners. *Empir Software Eng 26, 1*.

[Rishita Mullapudi 2021] Rishita Mullapudi, Tajmilur Rahman, J. N. (2021). Predicting software quality from development and release factors. In *IMMM 2021 : The Eleventh International Conference on Advances in Information Mining and Management*.

[Siegmund et al. 2020] Siegmund, N., Ruckel, N., and Siegmund, J. (2020). Dimensions of software configuration: on the configuration context in modern software development. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 338–349.