SOFTWARE PARA OTIMIZAR GERAÇÃO DE COMBINAÇÕES UTILIZANDO O MÉTODO *PAIRWISE*

Giovani França Sarchesi[1], Débora Pelicano Diniz[1]

[1]Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)

Ribeirão Preto, SP – Brasil

giovani.sarchesi@fatec.sp.gov.br,
debora.diniz2@fatec.sp.gov.br

Resumo. *Este artigo descreve o desenvolvimento de um sistema que auxilie a atividade de testes Caixa Preta, gerando casos de testes considerando o método de todos os pares, otimizando o tempo de criação das combinações de acordo com parâmetros informados pelo usuário. O Sistema foi desenvolvido utilizando C# e Excel para o armazenamento das combinações geradas.*

Abstract. *This article describes the development of a system that supports the activity of Black Box tests, generating test cases considering the method of all pairs, optimizing the creation time of the combinations according to the user parameters. The system was developed using C # and Excel to store the generated data.*

1. Introdução

Com os softwares cada vez mais robustos e complexos, o processo de teste vem tornando-se peça fundamental na Engenharia de Software, fazendo com que as responsabilidades de quem atua na área cresçam, pois, "teste é um elemento crítico para a garantia da qualidade de sistemas" (PRESSMAN, 2002).

Segundo Myers (1979 apud DELAMORO, 2016) teste é utilizar o programa com o intuito de encontrar erros e, para encontrar tais erros, são executados os seguintes passos: 1) montagem do conjunto de casos de testes, 2) execução do programa X, o submetendo ao conjunto criado, 3) análise do comportamento do programa X para saber se está de acordo com o especificado ou não. Repetindo os passos quantas vezes forem necessárias para chegar a um nível de confiança de que X atenderá seus usuários com o mínimo de erro possível.

A grande demanda que softwares recebem atualmente, levando em conta principalmente a cobrança sobre sua qualidade, tornou indispensável que, no processo de desenvolvimento de software, sejam adotados métodos, técnicas e ferramentas que permitam a realização da atividade de teste de maneira sistematizada e com fundamentação científica, de modo a aumentar a produtividade e a qualidade e a diminuir custos (DELAMORO, 2007, p.7).

Segundo Preto (2007), os *bugs* mais comuns em um programa geralmente são acionados por um único parâmetro de entrada ou por uma interação entre pares de parâmetros. Para esses casos, pode-se utilizar o método ***All-pairs Testing*** ou ***Pairwise*** (teste de todos os pares ou teste de pares) para definir os casos de testes. A técnica de teste de todos os pares é um método combinatório de teste de software que, para cada par de parâmetros de entrada para um sistema (normalmente, um algoritmo de software), testa todas as combinações discretas possíveis desses parâmetros. Usando

vetores de teste cuidadosamente escolhidos, isso pode ser feito muito mais rápido do que uma pesquisa exaustiva de todas as combinações de todos os parâmetros, paralelizando os testes de pares de parâmetros.

A técnica de teste de pares pode reduzir drasticamente o número de combinações a serem cobertas, mas permanece muito eficaz em termos de detecção de falhas. Na verdade, é uma técnica de design de teste inteligente que garante uma situação ganha-ganha, tanto para o esforço quanto para a eficácia do teste.

Mas, utilizar essa técnica em testes manuais é um trabalho demorado, assim, faz-se necessário o uso de algum software que auxilie nessa etapa de criação de casos de testes.

Assim, o objetivo do trabalho aqui apresentado é desenvolver um software que utilize o método de teste de todos os pares, procurando otimizar o tempo de criação das combinações de acordo com parâmetros informados pelo usuário.

2. Testes de software

Quando é dito que os testes são importantes, não se trata apenas da qualidade do software em si, mas também de custos que um teste ruim pode gerar.

Quanto mais tarde um defeito for identificado, mais caro fica para corrigi-lo, como pode ser observado no Gráfico 1. Os custos de descobrir e corrigir defeitos no software aumentam exponencialmente na proporção em que o trabalho evolui nas fases do projeto de desenvolvimento. Um outro aspecto que se deve considerar é o papel dos testes na manutenção dos sistemas. Uma grande parcela do orçamento de TI das organizações é dedicada à manutenção dos softwares após eles entrarem em produção (RIOS, 2013, p.14-15).

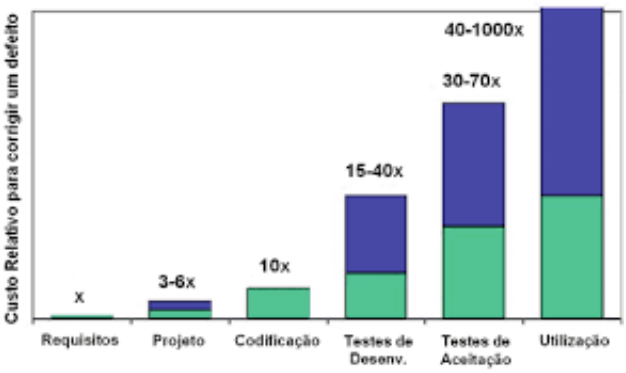


Gráfico 1 - Custo relativo para corrigir um defeito.
Fonte: Adaptado de (BOEHM, 1981)

2.1. Processo de Testes

O processo de testes deve basear-se em uma metodologia aderente ao processo de desenvolvimento, com pessoal técnico qualificado, em um ambiente adequado e utilizando ferramentas adequadas.

A metodologia de testes deve ser o documento básico para organizar a atividade de testar aplicações no contexto da empresa.

Como o desenvolvimento de sistemas é indesejável sem uma metodologia adequada, acontece o mesmo com a atividade de teste (RIOS, 2013, p.11). Na Figura 1 estão apresentadas as fases de um processo de testes. Pode-se verificar que o processo

de teste é composto por diferentes etapas, sendo quatro delas sequenciais e duas paralelas. De acordo com Rios (2006), o primeiro P (Procedimentos iniciais) representa uma etapa curta, na qual é traçado um pequeno esboço e assinado um acordo de nível de serviço, enquanto os outros dois P's (Planejamento e Preparação) acompanham todas as fases. Já as etapas mais importantes do processo de teste estão nos três E's (Especificação, Execução e Entrega), consumindo 80% a 85% do tempo no processo.

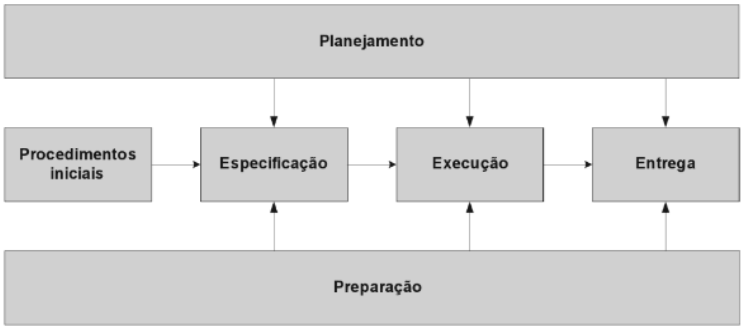


Figura 1 - Fases de um processo de testes.
Fonte: (RIOS, 2006)

2.2. Técnica de Testes de Caixa Preta

O Teste de Caixa-Preta é o teste baseado nos requisitos funcionais do software, ou seja, as condições de entrada dos casos de testes são definidas de forma a executar todos os requisitos funcionais especificados do sistema (PRESMAN, 2016).

A abordagem mais comum para esse tipo de teste, é escolher um subconjunto de entradas que maximize a riqueza dos testes, entradas essas que são definidas de acordo com o quê foi especificado como possíveis entradas para o software.

Ao utilizar essa técnica, o único interesse do testador não é em como os dados de entrada são processados, mas sim, se as saídas são coerentes com as entradas dadas, por exemplo, em um sistema que some valores, caso seja informado 1 e 1, para comprovar o funcionamento da funcionalidade, o software deve retornar 2.

2.3. Método Pairwise

Durante a fase de planejamento de teste de software, a técnica de teste ***Pairwise*** deve sempre ser levada em consideração. Seja manualmente ou utilizando qualquer ferramenta para gerar casos de teste, torna-se um componente necessário do plano de teste porque afeta a estimativa de teste.

A técnica de ***Pairwise*** é influenciada pelo procedimento matemático chamado matriz organizacional, que é formado por linhas e colunas, em que cada coluna representa uma variável (parâmetro do sistema) e cada linha representa uma possível combinação entre as variáveis e seus possíveis valores.

Para compreender melhor a técnica, Roden (2007) definiu 5 etapas para geração de casos de testes utilizando matrizes organizacionais: 1) identificar cada variável dentro do cenário desejado; 2) determinar o número de opções existentes para cada varável; 3) escolher o melhor enquadramento em uma matriz organizacional, levando em consideração que cada linha é um caso de teste, cada coluna uma variável de sistema e que em cada célula deve ser preenchida com as opções existentes em cada variável; 4) preencher a matriz, utilizando as opções possíveis para cada variável existente; 5)

especificar em cada linha um caso de teste que deverá ser executado conforme os valores atribuídos de cada variável.

Os testes são projetados de forma que, para cada par de parâmetros de entrada de um sistema, haja todas as combinações discretas possíveis desses parâmetros. O conjunto de testes cobre todas as combinações; portanto, não é exaustivo, mas muito eficaz na localização de *bugs*.

Na Figura 2 pode-se observar um exemplo de definição de parâmetros para ser usado no Método *Pairwise* no software *All-Pairs* e na Figura 3 está mostrado o resultado gerado pela aplicação. Observa-se na Figura 2, os nomes das variáveis na primeira linha, formando uma coluna para cada variável e abaixo de cada uma, seus respectivos valores possíveis. Após a listagem de todos, são geradas as combinações dos valores para encontrar todas as combinações, de acordo com os parâmetros e seus valores, conforme é demonstrado na Figura 3. Em alguns casos, nos quais a regra de negócio é complexa e existe uma grande quantidade de variáveis e valores possíveis, um número exorbitante de combinações pode ser encontrado, o que torna o processo maçante quando feito manualmente e de difícil compreensão quando utilizados softwares que não possuem interface gráfica como o *All-Pairs*.

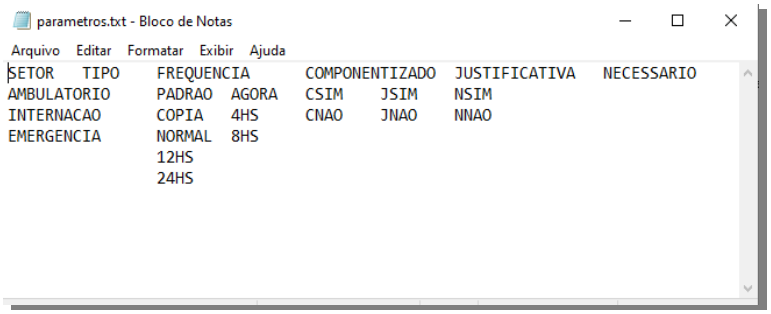


Figura 2 - Definição de parâmetros para usar All-Pairs
Fonte: Autoria própria

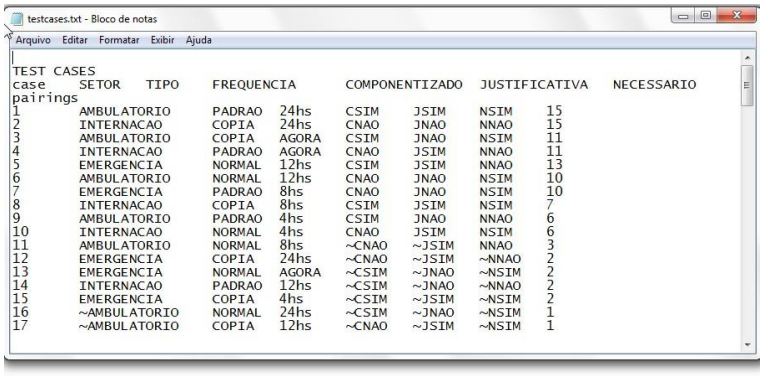


Figura 3 - Resultado gerado pela aplicação
Fonte: Autoria própria

3. Ferramentas utilizadas

Nessa seção serão apresentadas as ferramentas que foram utilizadas para o desenvolvimento do projeto.

3.1. C#

Para o desenvolvimento do *back-end* do software foi utilizada a linguagem C#, que é

uma linguagem de programação orientada a objetos e dirigida por eventos. C# pertence à família da linguagem C e adaptou os melhores recursos de C, C++ e Java, possuindo recursos próprios (HEJLSBERG ET AL, 2010).

Ela foi selecionada por permitir, de maneira relativamente fácil, a interação com softwares de diferentes linguagens e até ferramentas externas, como por exemplo o Excel, que também foi utilizado nesse trabalho. Além disso, os aplicativos C# podem interagir pela Internet usando padrões do setor, como SOAP (*Simple Object Access Protocol* - Protocolo Simples de Acesso a Objetos) e XML (*eXtensible Markup Language* - Linguagem de Marcação Extensível).

O C# possui acesso ao *.Net Framework Class Library*, uma coleção extensa de classes que permite amis rapidez no processo de desenvolvimento. Possui diversas funcionalidades como: portabilidade para diversas plataformas, *frameworks* para sistemas Web, recursos para programação assíncrona. O C# foi projetado para rodar sobre a plataforma Microsoft.Net com o .Net Framework (DEITEL & DEITEL, 2016).

Atualmente a linguagem está na versão 8.0.

3.2. *Windows Form*

O *front-end* do software foi desenvolvido utilizando o aplicativo *Windows Form*, que é orientado a eventos. O objetivo do *Windows Forms* é facilitar o desenvolvimento de interfaces gráficas em ambientes Windows, pois pode-se criar formulários utilizando os elementos gráficos nativos do Windows (NATHAN, 2007).

Ao contrário de um programa em lote, ele passa a maior parte do tempo aguardando que o usuário realize algo, como o preenchimento de um campo de texto ou o clique em um botão (WIKIPEDIA, 2021).

Todos os elementos visuais da biblioteca de classes do *Windows Form* são derivados da classe *Control*, isso fornece as propriedades básicas de um elemento de interface do usuário, como localização, tamanho, cor, fonte, texto, bem como eventos comuns, como clicar e arrastar/soltar.

3.3. Excel

Para guardar as combinações que serão geradas para o usuário foi utilizado o Excel, que é um editor de planilhas, lançado para o Mac em 1985 e sua primeira versão para Windows foi lançada apenas em novembro de 1987 e atualmente está na versão 16, conhecido como Excel 16 (MEYER, 2013).

No início, já existia um pacote de software chamado “Excel” na indústria financeira, o que fez o Excel ser alvo de um processo judicial, fazendo com que ele fosse chamado por muito tempo de “Microsoft Excel”, mas após algum tempo, isso foi sendo ignorado, e a questão foi resolvida quando a Microsoft adquiriu a marca registrada que era reservada ao outro programa (BRAGA, 2008).

A extensão de arquivo utilizada pelo Excel a partir da versão 12 é a “.xlsx”, mas anteriormente era a “.xls”.

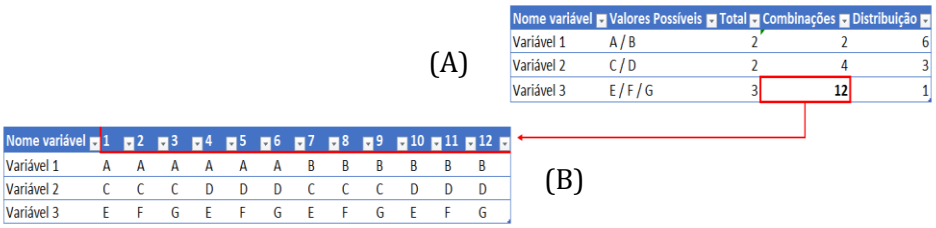
4. O sistema

Nessa seção serão demonstrados os escopos de documentação de Engenharia de Software utilizados para desenvolvimento do projeto. Todos os documentos produzidos durante o desenvolvimento estão disponíveis para consulta no GitHub, podendo ser acessado pelo link: https://github.com/giovani-sarchesi/TCC.

4.1. Requisitos do Sistema

O principal objetivo do software é auxiliar na criação de combinações, utilizando o método ***All-Pairs*** ou ***Pairwise***, otimizando o tempo gasto para aplicação da técnica.

A principal função do sistema é gerar e gravar as combinações, utilizando os parâmetros informados pelo usuário, ou seja, as variáveis necessárias para o teste. Deve ser informado o nome da variável e seus valores possíveis, como pode ser observado na Figura 4, Tabela A. Os cálculos para as combinações serão executados multiplicando os valores possíveis das variáveis de cima para baixo, para verificar quantas combinações devem ser feitas (os valores sendo armazenados na coluna **Combinações,** da Tabela A da Figura 4), e depois dividindo de baixo para cima os valores obtidos anteriormente, para verificar quantas vezes cada valor das variáveis devem ser repetidos (com os valores sendo armazenados na coluna **Distribuição,** da Tabela A da Figura 4). Na Figura 4 Tabela B está apresentado um exemplo de geração de combinações.



(A)

Nome variável	Valores Possíveis	Total	Combinações	Distribuição
Variável 1	A / B	2	2	6
Variável 2	C / D	2	4	3
Variável 3	E / F / G	3	12	1

(B)

Nome variável	1	2	3	4	5	6	7	8	9	10	11	12
Variável 1	A	A	A	A	A	A	B	B	B	B	B	B
Variável 2	C	C	C	D	D	D	C	C	C	D	D	D
Variável 3	E	F	G	E	F	G	E	F	G	E	F	G

Figura 4 – Exemplo montagem manual
Fonte: Autoria própria

Depois de informar todas as variáveis necessárias para o teste, o usuário informará o nome que deseja para o arquivo Excel (.xlsx) que será gerado e todos serão exportados para o caminho "C:\Combinacoes". Um tratamento para verificar se já existe um arquivo com o nome informado foi feito, e o usuário poderá decidir se substitui os dados antigos ou informa outro nome para o novo arquivo.

4.2. Execução

A primeira tela a ser mostrada ao executar o software é a de entrada de dados (Figura 5), na qual o usuário deve entrar o nome da variável e seus possíveis valores.

Nessa tela são apresentadas também todas as variáveis já informadas. Após incluir todas as variáveis, deve-se informar o nome do arquivo em que serão arquivados os casos de testes e, por fim, apertar o botão **Gerar Combinação.** Na Figura 6 está apresentada a tela inicial com as informações sobre as variáveis preenchidas e na Figura 7 está apresentada a tela com a informação sobre o nome do arquivo preenchida.

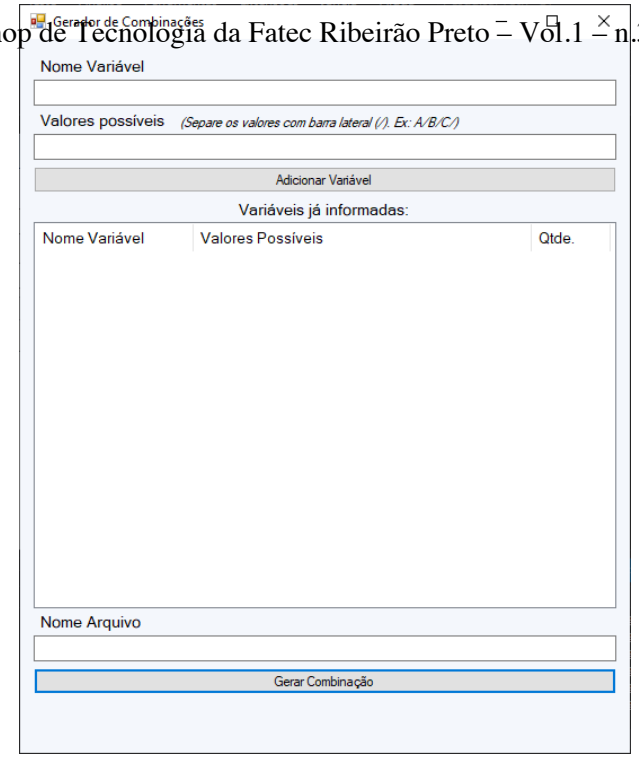


Figura 5 - Tela principal do software.
Fonte: Autoria própria

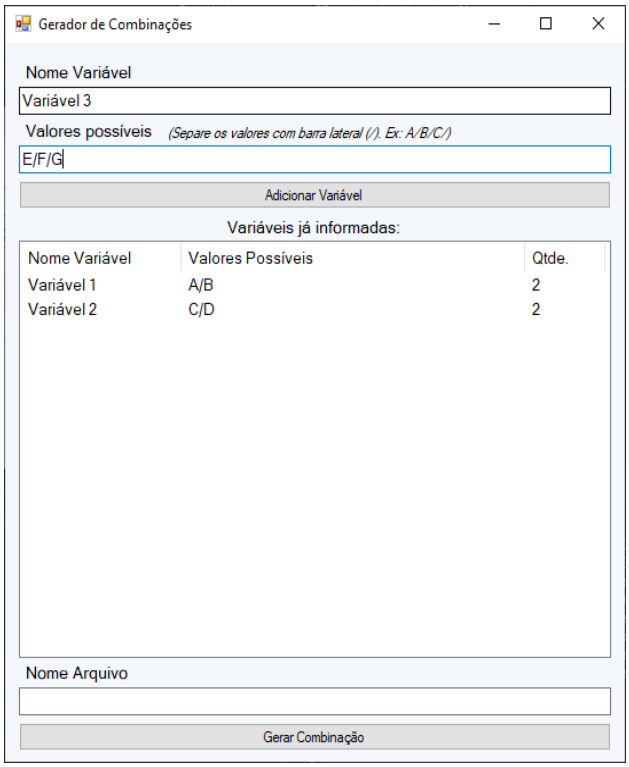


Figura 6 - Variáveis adicionadas
Fonte: Autoria própria

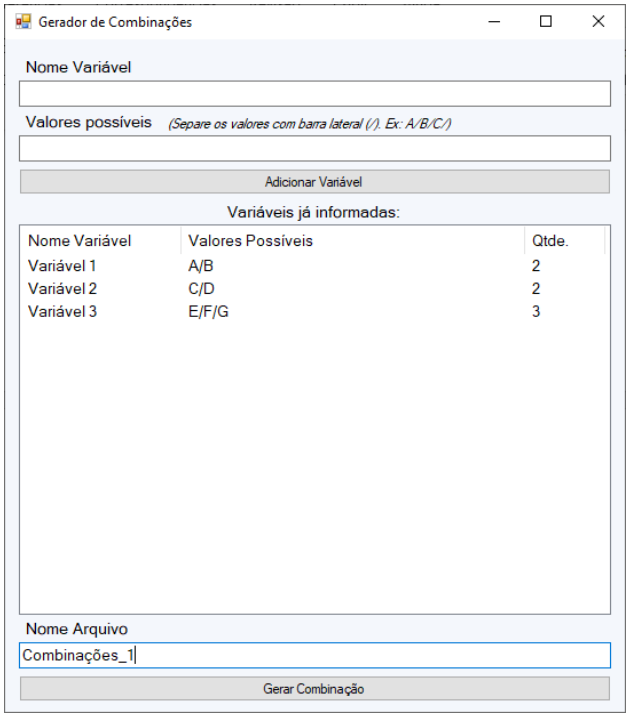


Figura 7 - Informando nome para o arquivo
Fonte: Autoria própria

Após clicar no botão Gerar Combinação, os casos de testes são gerados, considerando o método **All Pairs**, e todas as combinações são gravadas em um arquivo do Excel, com o nome do arquivo sendo o entrado pelo usuário. Ao Finalizar o processo, a mensagem **'Arquivo gerado com Sucesso'** aparece na tela, com a informação da pasta na qual está armazenado o arquivo, como pode ser observado na Figura 8.

Na Figura 9 está apresentado o arquivo Excel gerado com as combinações do exemplo utilizado.

Importante comentar que não foram implementadas funcionalidades para edição e exclusão dos dados das variáveis.

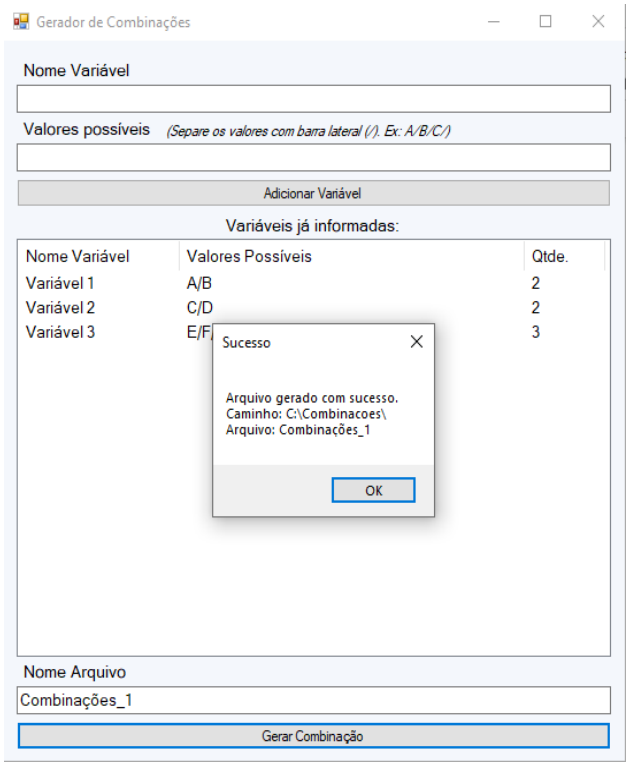


Figura 8 - Confirmação da criação do arquivo
Fonte: Autoria própria

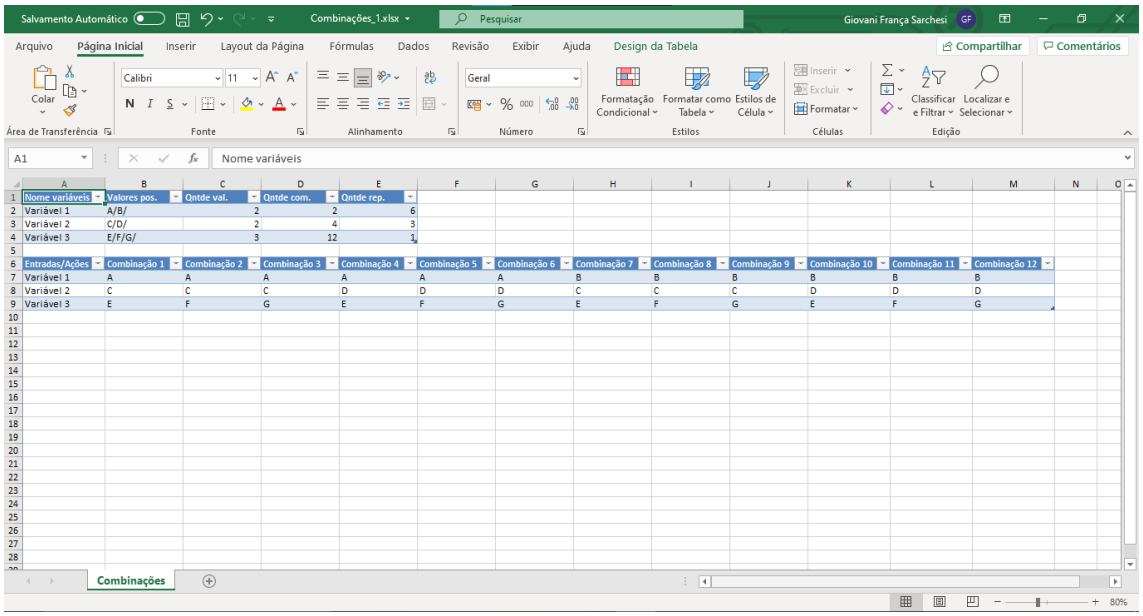


Figura 9 - Arquivo gerado com as combinações
Fonte: Autoria própria

4.3. Teste comparativo

Para validar o software desenvolvido no projeto foi feita uma comparação entre o tempo que o software gasta para gerar o arquivo de combinações e o tempo necessário para montar as combinações de forma manual. Para tal teste foi realizada simulação com variáveis que envolvem uma compra em um *ecommerce*.

As variáveis e valores utilizados no teste foram:

- Produtos no carrinho – Sim/Não
- Cupom – Sim/Não
- Frete – Gratuito/Pago
- Brinde – Sim/Não
- Forma de pagamento – Dinheiro/Cartão Débito/Cartão Crédito/Boleto
- Para presente – Sim/Não

Com essas variáveis, o número de combinações necessárias utilizando o método **All Pairs** foram 128 (Figura 10), o software gastou 2 segundos para gerar o arquivo (Figura 11), já de forma manual, foram gastos aproximadamente 6 minutos.

Ambos os arquivos estão no repositório citado na seção 4.

Nome variáveis	Valores pos.	Qntde val.	Qntde com.	Qntde rep.
Produto no carrinho	Sim/Não/	2	2	64
Cupom	Sim/Não/	2	4	32
Frete	Gratuito/Pago/	2	8	16
Brinde	Sim/Não/	2	16	8
Forma de pag	Dinheiro/Cartão Débi/Cartão Créd/Boleto/	4	64	2
Para presente	Sim/Não/	2	128	1

Figura 10 - Cabeçalho teste
Fonte: Autoria própria

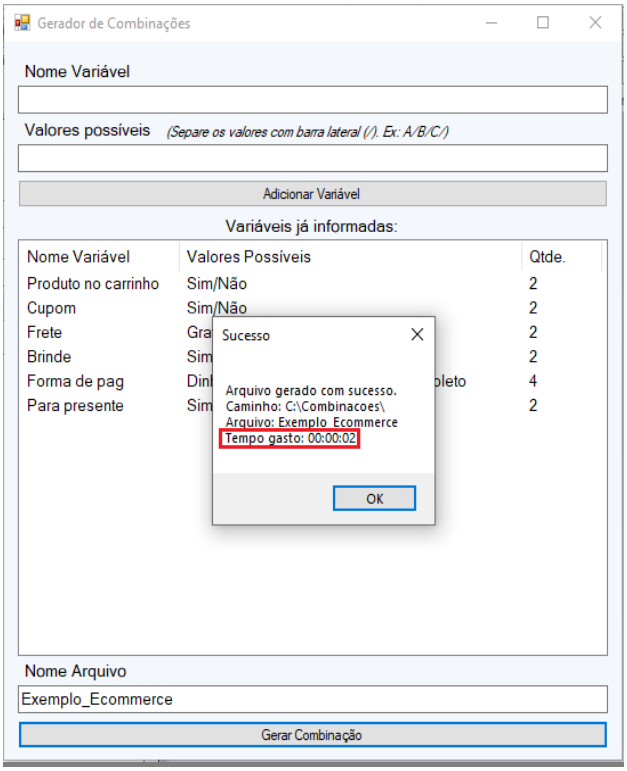


Figura 11 - Tempo gasto para gerar arquivo
Fonte: Autoria própria

5. Conclusão

A evolução tecnológica aumentou a demanda por técnicas e ferramentas que abordam o problema da qualidade dos softwares e é possível encontrar na literatura diferentes estratégias e técnicas de testes que auxiliam a garantir a qualidade do sistema, bem como ferramentas que auxiliam no processo de definir os casos de testes.

O objetivo do trabalho apresentado neste artigo foi o desenvolvimento de uma ferramenta que automatize a geração de casos de testes, utilizando o método de teste de todos os pares, procurando otimizar o tempo de criação das combinações de acordo com parâmetros informados pelo usuário. A ferramenta foi construída e está funcionando adequadamente, visto que em todos os testes realizados, as combinações foram corretamente criadas e o tempo gasto pelo software para gerar as combinações foi menor que o necessário para gerar as combinações de forma manual.

Como trabalhos futuros, são sugeridos:

- implementar funcionalidades para edição e exclusão dos dados das variáveis, tornando o software mais completo, visto que, no momento, o software permite que o usuário apenas adicione variáveis e seus valores possíveis;
- implementar funcionalidade para que o próprio software defina quais são as melhores combinações para o teste, visto que, atualmente, o software exporta todas as combinações possíveis para o arquivo ".xlsx".

Referências

BRAGA, J. C. P. (2008). O Uso da Planilha Eletrônica Como Ferramenta na Matemática do Ensino Médio do Centro Federal de Educação Tecnológica de JanuáriaMG. Seropédica, UFRRJ,RJ. 91 p. (Dissertação, Mestrado em Educação Agrícola).

BOEHM, B.W. (1981). *Software Engineering Economics.* Prentice Hall.

DEITEL, H., & DEITEL, P. (2016). *C# 6 for Programmers* (6th ed.). Prentice Hall.

DELAMORO, Márcio Eduardo. Introdução ao teste de software. 1. ed. São Paulo: Elservie, 2007.

DELAMORO, Márcio Eduardo. Introdução ao teste de software. 2. ed. São Paulo: Elsevier, 2016.

HEJLSBERG, A., GOLDE, P., WILTAMUTH, S., & TORGERSEN, M. (2010). *The C# Programming Language* (4th ed.). Addison-Wesley Professional

MEYER, M. (2013). O que é excel? Disponível em: <https://www.aprenderexcel.com.br>. Acesso em 14 mai. 2021.

NATHAN, A. *Windows Presentation Foundation Unleashed. Sams Publishing*, 2007.

PRETO, R. Teste de software pragmático: se tornando um profissional de teste eficaz e eficiente. Nova York: WILLEY., 2007.

PRESSMAN, R. S. Engenharia de Software. 5. ed. Rio de Janeiro: McGraw-Hill, 2002.

PRESSMAN, R. Engenharia de Software – Uma abordagem profissional. 8. ed. Porto Alegre: AMGH EDITORA LTDA, 2016.

RIOS, E. Teste de Software. 2 ed. Rio de Janeiro: Alta Books, 2006.

RIOS, E. Teste de Software. 3 ed. Rio de Janeiro: Alta Books, 2013.

RODEN, L. *Pairwise Testing. An Easy Guide to Orthogonal Arrays & All-pairs Combinations*. Better Software Magazine, 2007.

WINDOWS FORM. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2021. Disponível em: < https://en.wikipedia.org/w/index.php?title=Windows_Forms&oldid=1012077315>. Acesso em: 14 mar. 2021.