Ferramenta de medição de Worst-Case Execution Time (WCET) utilizando o Algoritmo de Otimização por Colônia de Formigas (ACO) para Software Embarcado de Uso Aeronáutico

Bruno R. Aricó[1,2], Elpidio C. de. A. Bisneto[1,2], Guilherme I. Melaninho[1,2], Gustavo G. O. Pereira[1,2], Iraline N. Simões[1,2], Edna N. da S. Barros[1], Renzo de C. Junior[3]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)

[2]Programa de Especialização em Software Software Embarcado

[3]Embraer

Abstract. *This article presents an innovative tool for measuring Worst Case Execution Time (WCET) in embedded software for aeronautical use. Using C language algorithms, the proposed tool calculates the WCET using the Ant Colony Optimization Algorithm (ACO), conventional methods, such as IPET, and dynamic measurements in hardware. In this context, for each algorithm proposed in the testbench, a comparison is made based on the value obtained by each method. The results indicate that the ACO approaches the IPET values, demonstrating its effectiveness in estimating the WCET. Despite presenting greater variability compared to direct measurements, ACO stands out as a promising alternative for calculating WCET in aeronautical embedded systems.*

Resumo. *Este artigo apresenta uma ferramenta inovadora para medição do Tempo de Execução no Pior Caso (WCET) em software embarcado de uso aeronáutico. A partir de algoritmos em linguagem C, a ferramenta proposta realiza o cálculo do WCET utilizando o Algoritmo de Otimização por Colônia de Formigas (ACO), métodos convencionais, como o IPET, e medições dinâmicas em hardware. Nesse âmbito, para cada algoritmo proposto no testbench, é feita uma comparação a partir do valor obtido por cada método. Os resultados indicam que o ACO se aproxima dos valores do IPET, evidenciando sua eficácia na estimativa do WCET. Apesar de apresentar maior variabilidade em comparação com medições diretas, o ACO destaca-se como uma alternativa promissora para o cálculo do WCET em sistemas embarcados aeronáuticos.*

1. Introdução

Sistemas de tempo real são essenciais para o funcionamento de muitas aplicações modernas, sendo aplicados em diversas áreas como sistemas ferroviários, aeronáuticos, automotivo, telecomunicações, biomedicina, entre outros [Bygde et al. 2009]. Em ferrovias, um sistema de tempo real garante que os trens cheguem na hora certa e que não colidam uns com os outros. Em um carro, um sistema de tempo real pode garantir que os freios funcionem corretamente quando são acionados. Em um aparelho biomédico, um sistema de tempo real pode garantir que os dados dos pacientes sejam coletados e analisados de forma precisa [Kumar 2022].

O número de sistemas embarcados em aeronaves vem aumentando significativamente e incorporam vários sistemas, incluindo sistemas que são críticos para a segurança do voo [SAV]. Esses sistemas são comumente categorizados como *hard real-time* [Jean et al. 2015]. Portanto, as tarefas a serem executadas devem atender aos seus compromissos de tempo dentro de intervalos predefinidos (*deadlines*) para que não ocorra uma condição de falha. O Tempo Máximo de Execução (WCET, do inglês *Worst-Case Execution Time*) é uma métrica crucial na avaliação do desempenho de sistemas em tempo real [Silva et al. 2019].

O cálculo do WCET deve ser seguro, ou seja, deve ser preciso para não ocorrer a subestimação do tempo de execução e eficaz para minimizar superestimações significativas. Isto se deve ao fato de que, no contexto de escalonamento de sistemas de tempo real, é necessário que as restrições temporais sejam mapeadas em termos destes prazos. O escalonamento de forma geral é um problema combinatório e difícil de resolver, pertencendo à classe de problemas NP-difícil, conforme a teoria da complexidade computacional, o que torna as análises ainda mais complexas [Silva et al. 2015].

Alguns elementos do processador como implementação em *pipeline*, memórias *cache* e sistemas operacionais (SO), podem influenciar no tempo de execução de um código. Para estimar o WCET existem três métodos principais [Benedicte et al. 2016, Ballabriga et al. 2015]. A análise estática, que é uma abordagem mais analítica e precisa, envolve uma análise mais detalhada do código, porém pode implicar em grande custo computacional e de tempo. A análise de medição consiste na execução da aplicação no hardware ou em um simulador para medir o tempo de execução. A vantagem desse método é a redução de recursos computacionais, porém exige análises mais rígidas, incluindo dados de entrada e cenários de execução. Há também o método híbrido, que combina a análise estática e a análise por medição [de Oliveira 2020].

Para calcular analiticamente o pior caminho de execução (WCEP, do inglês *Worst-Case Execution Path*), os métodos mais utilizados incluem a técnica de enumeração implícita de caminhos (IPET, do inglês *Implicit Path Enumeration Tecnique*) e o método baseado em caminhos (*path-based*). Em determinados casos, a busca pelo pior caminho pode ser muito difícil devido à complexidade e ao tempo computacional [Ermedahl et al. 2005].

Uma abordagem promissora para realizar o cálculo do WCET é a aplicação de algoritmos bioinspirados, os quais se inspiram em princípios e padrões observados na natureza. Estes algoritmos têm demonstrado eficácia em resolver problemas complexos e otimizar processos em diversas áreas. Entre esses algoritmos, o algoritmo de Otimização por Colônia de Formigas (ACO) se destaca [Dorigo et al. 2006, Silva et al. 2019]. Inspirado no comportamento das formigas em busca de caminhos mais eficientes entre a fonte de alimento e o formigueiro, o ACO é capaz de encontrar soluções ótimas em problemas complexos e pode ser um candidato promissor para realizar o cálculo do WCET.

Neste trabalho, propomos o desenvolvimento de uma ferramenta de cálculo do WCET utilizando três métodos distintos: o método IPET, o algoritmo ACO e a medição dinâmica no hardware, realizando comparações de desempenho entre eles. O restante do artigo está estruturado da seguinte forma: a Seção 2 apresenta a justificativa do projeto, a Seção 3 descreve o referencial teórico, apresentando os métodos de cálculo do WCET, os trabalhos relacionados e uma descrição do algoritmo ACO. A Seção 4 apresenta a proposta de

projeto, enquanto a Seção 5 detalha o desenvolvimento do orquestrador, a adaptação do ACO para o cálculo do WCET e a implementação do cálculo dinâmico. Os resultados são expostos na Seção 6 , enquanto as Seções 7 e 8 abordam a conclusão e os futuros trabalhos, considerando as Seções 4, 5 e 6.

2. Justificativa

A escolha do algoritmo ACO em detrimento de métodos convencionais, como o IPET, para calcular o WCET, pode ser justificada por diversas razões. O ACO oferece uma abordagem não convencional, inspirada no comportamento de formigas, permitindo explorar soluções de forma inovadora. Sua capacidade de adaptação dinâmica a alterações no ambiente e a busca estocástica em espaços de solução complexos fazem dele uma escolha robusta em situações onde as condições de execução podem variar. Além disso, a natureza colaborativa do ACO, combinada com sua capacidade de paralelismo e distribuição, o torna adequado para problemas em que a eficiência na busca por soluções é crucial, como é o caso do cálculo do WCET em sistemas complexos.

3. Referencial Teórico

3.1. Cálculo do WCET

O WCET consiste no tempo de execução de uma tarefa em seu pior caso. É um conceito muito importante para sistemas de tempo real críticos, visto que as tarefas devem cumprir um *deadline* [Kebbal 2006]. Porém, existem diversos fatores relacionados aos processadores que tornam desafiador o cálculo do WCET para sistemas de tempo real, como *pipeline*, memória *cache*, *branch prediction*, entre outros [Silva et al. 2015]. O cálculo do WCET é utilizado para obter o pior tempo de execução das tarefas e, a partir destas informações, avaliar a viabilidade do escalonamento do sistema [Lv et al. 2009].

A análise do WCET pode ser categorizada em 3 tipos: a dinâmica, a estática e a híbrida. A abordagem dinâmica consiste em calcular o tempo de execução utilizando o hardware proposto ou um simulador com acurácia de tempo. Desta forma, são feitas várias medições para se ter um valor máximo, e este valor máximo é dado como o WCET. A análise estática é aquela em que a avaliação é conduzida a partir de um modelo de hardware proposto, sem a execução efetiva do algoritmo [Lv et al. 2009]. O modelo híbrido de análise combina o melhor de cada uma das duas outras técnicas, incorporando uma análise estática que guia a simulação do algoritmo no hardware proposto [Huybrechts et al. 2018].

Outra forma de classificar os métodos de cálculo do WCET é entre determinístico e probabilístico. Desta forma, dado um modelo de hardware e um algoritmo de software, o resultado do WCET calculado por uma ferramenta determinística deverá ser sempre o mesmo. Já as ferramentas probabilísticas podem ter um comportamento diferente ao longo do tempo para a mesma entrada [Abella et al. 2015].

Uma análise estática do WCET pode ser dividida em 3 partes principais: a análise do fluxo do algoritmo, uma análise de baixo nível e, por fim, o cálculo do WCET em si [Ermedahl et al. 2005]. Na análise de fluxo do algoritmo, é gerado um grafo anotado com os tempos de execução que representam o comportamento do mesmo. A partir deste

grafo, existem diversas formas de resolvê-lo. O modo mais intuitivo é o método baseado em caminhos, onde são testados todos os caminhos possíveis de um grafo para ver qual é o caminho com pior tempo de execução. Outra forma de resolver é aplicando a técnica baseada em ramificações, onde, em cada ramificação, é feita a substituição desta ramificação por um bloco simples [Ermedahl et al. 2005].

Porém, o método mais utilizado para a resolução do grafo de fluxo de controle é o IPET. O IPET assume que cada vértice do grafo tem um custo associado e, com base nesses custos, são criadas restrições lineares para excluir caminhos impossíveis do grafo. A partir disso, é feito o cálculo do custo máximo de circulação pelo grafo. Desta forma, o IPET consiste em modelar a aplicação em um problema ILP (do inglês *Integer Linear Problem*) [Huber and Schoeberl 2009].

Além destes métodos de resolução do WCET, existem outras técnicas sendo desenvolvidas para aprimorar a precisão e eficiência do cálculo. Uma abordagem mais recente para enfrentar este desafio será apresentada no próximo tópico.

3.2. Algoritmo Bioinspirado para WCET

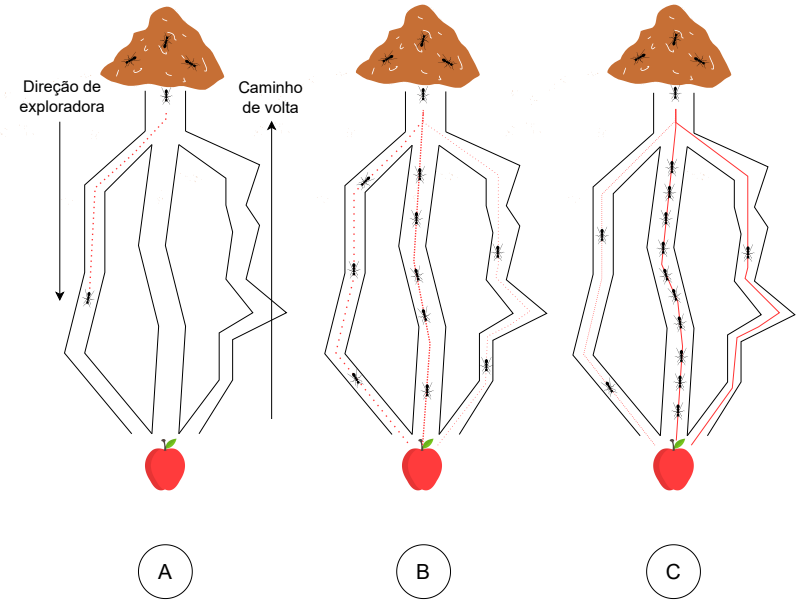
Algoritmos bioinspirados têm sido utilizados para encontrar conjuntos de dados de teste. O estudo de [Silva et al. 2019] utilizou algoritmo genético (GA) para identificar os melhores e piores valores de execução de um programa em *software* embarcado. Os estudos de [Wegener and Mueller 2001] utilizaram o GA para realizar testes em *hardware* para encontrar o pior tempo de execução. O trabalho de [Lee et al. 2023] utilizou GA combinado com aprendizagem de máquina. Neste contexto, o modelo preditivo foi utilizado para substituir o simulador de precisão para gerar entradas em um sistema de tempo real que simula o WCET e o GA foi utilizado para encontrar os dados de entrada que resultassem no pior tempo de execução.

3.3. ACO

O algoritmo de otimização de colônia de formigas (ACO, do inglês *Ant Colony Optimization*) é inspirado no comportamento de algumas espécies de formigas na busca por encontrar o menor caminho até o alimento. A formiga deposita feromônios no chão para demarcar um caminho favorável. As demais formigas são atraídas pela trilha de feromônios e provavelmente escolherá a trilha que tiver mais feromônios [Dorigo et al. 1996, Dorigo et al. 2006]. A Figura 1-A mostra como esse método funciona, onde a formiga sai da colônia em busca de alimento, em **B** as formigas exploram os diferentes caminhos aumentando a trilha de feromônios no menor caminho, por fim em **C** as formigas tendem a escolher o caminho com maior feromônio.

No contexto de otimização, as soluções são representadas como caminhos em um grafo, e o ACO é utilizado para encontrar o caminho mais curto ou a solução com o menor custo, dependendo do problema em questão. Cada caminho é associado a uma trilha de feromônios, que é atualizada ao longo das iterações. Em cada nó a formiga tomará a decisão por qual caminho seguir através da equação:

Figura 1. Exemplo de funcionamento do ACO, onde a formiga sai da colônia em busca de alimento em A, em B as formigas exploram os diferentes caminhos aumentando a trilha de feromônios no menor caminho, por fim em C as formigas tendem a escolher o caminho com maior feromônio.



$$P_{i,j} = \frac{(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}{\sum((\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta})} \tag{1}$$

em que $P_{i,j}$ é a probabilidade de uma formiga se mover do nó i para o nó j, $\tau_{i,j}$ é a trilha de feromônios na aresta de i para j e α e β controlam a importância relativa do feromônio e da informação heurística representado por $\eta_{i,j}$ em que:

$$\eta_{i,j} = \frac{1}{L_{i,j}} \tag{2}$$

em que $\frac{1}{L_{i,j}}$ é a distância entre i e j.

Para selecionar o próximo nó que a formiga irá se mover, pode ser utilizado o método de seleção da roleta. Ele consiste em colocar as probabilidades de cada caminho em uma roleta e, a partir da soma dessas probabilidades, é gerado um número aleatório. Assim, quando a soma das probabilidades dos caminhos for maior que este número aleatório, a formiga irá tomar este caminho [Behera 2020].

A cada iteração do algoritmo, os valores de feromônios são atualizados da seguinte forma:

$$\tau_{i,j}^{k} = (1-\rho)\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k} \tag{3}$$

onde ρ é a taxa de evaporação, m é o número total de formigas, $\Delta\tau_{i,j}$ é a quantidade de feromônio depositado no caminho de i para j pela formiga k,

$$\begin{cases} \frac{1}{L_k} & \text{se a formiga k usou o caminho (i,j) como seu melhor caminho} \\ 0 & \text{outros casos} \end{cases} \tag{4}$$

No algoritmo de [Stützle and Hoos 2000] foi feito uma melhoria do sistema em que a melhor formiga atualiza as trilhas de feromônio,

$$\tau_{i,j} \leftarrow \left[(1-\rho) \cdot \tau_{(i,j)} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{best} \right]_{\tau_{min}}^{\tau_{max}} \tag{5}$$

onde $\Delta\tau_{i,j}^{best}$ é a trilha depositada pela melhor formiga que usou o L_{best} que é o melhor caminho encontrado durante a iteração e τ_{max} e τ_{min} o limite de feromônios que pode ser depositado, esses valores são normalmente obtidos empiricamente ou em base de considerações analíticas [Stützle and Hoos 2000]. O APÊNDICE A mostra o fluxograma do código ACO em que se inicializa os parâmetros do algoritmo, em seguida é verificado se o número de iterações terminou, caso não tenha terminado é realizado o cálculo probabilístico e em seguido selecionado a rainha e atualizado a taxa de feromônio na matriz de feromônios. Se a solução da iteração atual for melhor que a anterior, atualiza a melhor solução global, caso o contrário permanece a solução anterior. Ao final de todo o ciclo de iterações é exibido o melhor valor global.

4. Proposta de Projeto

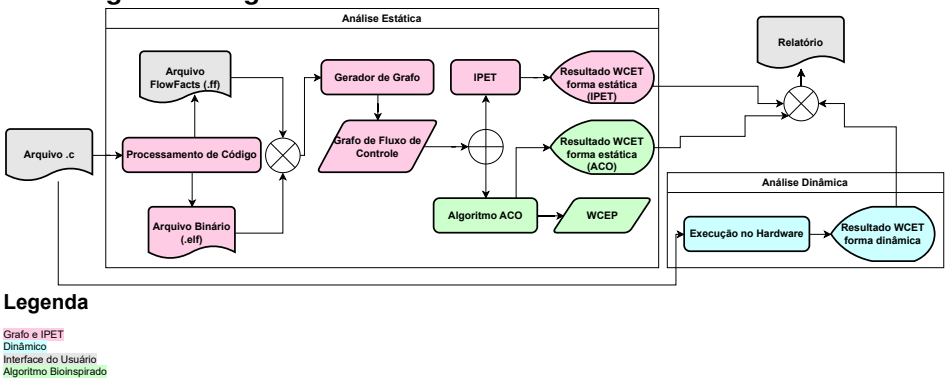
4.1. Descrição do sistema

O sistema proposto consiste em uma ferramenta de cálculo do WCET chamada BioWcet, desenvolvida majoritariamente em C++. A ferramenta recebe como entrada um código em linguagem C para realizar o cálculo do tempo de pior caso de execução através de três métodos distintos, exibindo os resultados na saída. O BioWcet utiliza o arquivo .c, fornecido como entrada, no framework OTAWA para gerar o grafo de fluxo de controle (CFG) e para exibir o valor de WCET com base no método IPET. O CFG gerado pelo OTAWA é transformado em uma matriz esparsa, que é utilizada como entrada do ACO. Além disso, o arquivo .c também é enviado ao microcontrolador para realizar o cálculo de forma dinâmica. A ferramenta está disponível em `https://github.com/gugagop/TCC-PES02` e o modelo do fluxo da solução proposta é apresentado na Figura 2.

Os principais blocos que compõe o fluxo são:

- **Arquivo .c :** Arquivo inserido pelo usuário que contém o código a ser analisado.
- **Processamento de código :** Nesta etapa ocorre a criação dos arquivos necessários para a geração do CFG. O arquivo de entrada é compilado de acordo com o modelo de arquitetura adotado e então, são gerados dois arquivos distintos. O primeiro é o arquivo binário (.elf), que contém a representação do código em binário. O segundo arquivo, chamado *FlowFacts* (.ff), contém informações sobre a quantidade de iterações que cada laço identificado no código deve realizar.
- **Gerador de Grafo :** A partir dos arquivos *.ff* e *.elf*, é criado o CFG com o tempo dos blocos básicos. O grafo facilita a análise do programa e a identificação de pontos críticos que podem afetar o WCET.

Figura 2. Diagrama de blocos do fluxo da ferramenta BioWcet.



- **IPET :** Realiza o cálculo WCET utilizando o método do IPET a partir do CFG.
- **Algoritmo ACO :** Algoritmo heurístico utilizado para encontrar o WCEP e calcular o WCET a partir do grafo.
- **Execução no Hardware :** O arquivo .c é compilado precisamente para o microcontrolador utilizado e executado diretamente no hardware.

4.2. Hardware utilizado

A análise dinâmica do projeto foi conduzida no microcontrolador Bluepill 32bits STM32F103C6T6, que apresenta o processador ARM Cortex-M3 e adota a arquitetura Armv7-M. Os microcontroladores ARM Cortex-M3 oferecem alta escalabilidade, alto desempenho em tempo real em aplicativos com custos limitados e podem realizar tarefas complexas. Ele possui um pipeline de 3 estágios para otimizar o processamento de instruções, suporta diversas ferramentas de desenvolvimento e oferece suporte ao conjunto de instruções Thumb-2. Essas características fazem dele uma escolha interessante para a execução do projeto em questão.

4.3. Requisitos do projeto

O levantamento e análise de requisitos são processos importantes para assegurar que o sistema seja desenvolvido com qualidade e atenda as necessidades ao qual o sistema foi projetado. Os requisitos do projeto foram divididos em requisitos funcionais e não funcionais. Todos os requisitos foram revisados por duas pessoas.

Os requisitos funcionais foram categorizados em dois níveis: requisitos funcionais de sistema e requisitos de alto nível. Os requisitos funcionais de sistema representam as funcionalidades específicas que o sistema deve fornecer. Já os requisitos de alto nível referem-se aos requisitos de software mais abstratos e fundamentais. Os requisitos não funcionais descrevem os critérios de qualidade que o sistema deve atender. A nomenclatura e a quantidade de cada tipo de requisito estão representadas na Tabela 1. Todos os requisitos se encontram no APÊNDICE C.

Tabela 1. Requisitos com suas nomenclaturas e suas quantidades

Tipo de Requisito	Nomenclatura	Quantidade Elaborada
Requisitos Funcionais de Sistema	FRS	16
Requisitos de Alto Nível Gerador Grafo	HRG	5
Requisitos de Alto Nível IPET	HRIT	1
Requisitos de Alto Nível Interface	HRI	14
Requisitos de Alto Nível Algoritmo BioInspirado	HRB	7
Requisitos de Alto Nível Dinâmico	HRD	5
Requisitos Não Funcionais	NFR	6

4.4. Restrições de projeto

Para o desenvolvimento do projeto, foram consideradas algumas restrições para que a ferramenta atenda os requisitos e tenha um bom desempenho. Algumas dessas restrições são:

- **OTAWA :** Para rodar o BioWcet é necessário ter a ferramenta OTAWA e suas dependências previamente instaladas e configuradas.
- **Ambiente :** O BioWcet foi projetado para rodar em sistemas Linux x64, necessitando de 50 *megabyte* (MB) de espaço em disco para ser instalado. Para facilitar a instalação do *software* e suas dependências, uma imagem *docker* da instalação está disponibilizada, com os passos de instalação e configuração. Com isso, BioWcet torna-se multiplataforma, uma vez que é possível utilizar *containers Docker* em ambientes *Windows* e *macOS*.
- **Arquivo de entrada :** O arquivo de entrada deverá ser na linguagem C e possuir a extensão .C, além disso o código não deve conter recursão, alocação dinâmica de memória e não deverá conter ponteiro.
- **Análise dinâmica :** Atualmente o BioWcet é compatível com a arquitetura ARM Cortex-M3.

4.5. Arquitetura do sistema

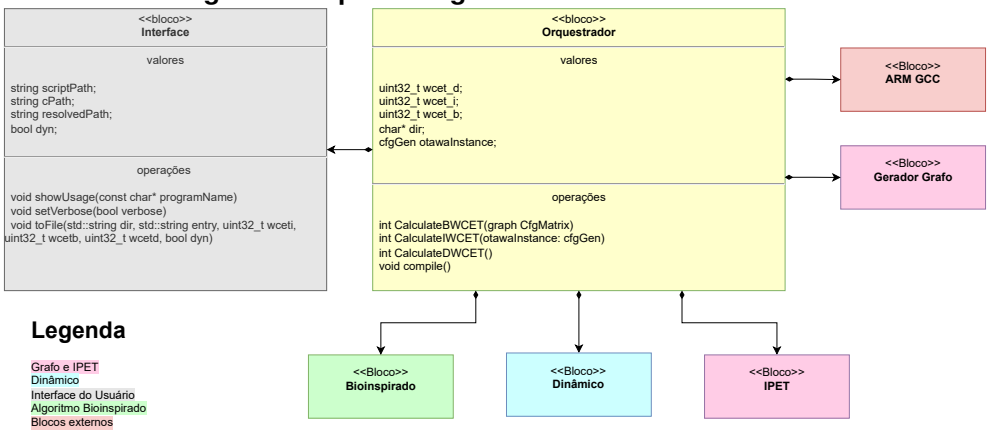
Diante dos requisitos levantados foi decidido adotar um padrão de projeto que conseguisse promover a modularidade e a reutilização de código, facilitando a manutenção, desenvolvimento ágil e a escalabilidade do sistema. Assim, foi escolhida a arquitetura modular, fazendo uso dos padrões de projeto fachada e mediador para garantir um sistema organizado e flexível.

O padrão fachada proporciona uma interface única e simplificada para interações entre diferentes módulos do sistema. Já o padrão mediador tem como objetivo simplificar a comunicação entre os componentes, reduzindo o acoplamento direto entre eles. Essas escolhas visam criar um desenvolvimento mais eficiente e garantir um sistema adaptável a futuras melhorias.

Os blocos gerais que compõem a arquitetura são:

- **Interface :** Responsável por proporcionar a interação do usuário com a solução.

Figura 3. Arquitetura geral da ferramenta BioWcet.



É através dela que o usuário fornece os dados de entrada da ferramenta, solicita a execução dos cálculos e visualiza os resultados obtidos.

- **Orquestrador :** Responsável por coordenar e controlar a interação entre os demais componentes do sistema. Atua como um elemento central que gerencia a execução das diferentes partes, garantindo um bom acoplamento entre todas as partes.
- **Bioinspirado :** Responsável por fornecer o WCEP e calcular o WCET utilizando o algoritmo ACO.
- **Dinâmico :** Responsável por fornecer o WCET utilizando uma análise dinâmica.
- **IPET :** Responsável por fornecer o valor do WCET utilizando o método IPET. Este bloco se beneficia da integração com o OTAWA, sendo uma interface para o método original fornecido pelo framework.
- **Gerador do Grafo :** Responsável por gerar o CFG do arquivo de entrada, este bloco se utiliza do CFG gerado pelo OTAWA, anotado com os ciclos consumidos por cada bloco básico, para gerar uma matriz de adjacências num formato esperado pelo algoritmo ACO.
- **ARM GCC :** Responsável por compilar o código fonte e criar binários executáveis.

4.6. Rastreabilidade

A rastreabilidade desempenhou um papel importante no controle de configuração do sistema desenvolvido, desde a criação dos requisitos até o desenvolvimento e execução dos testes. A divisão dos requisitos em camadas proporcionou uma visão clara de todos os elementos do projeto, facilitando o rastreamento entre os requisitos de sistema e os requisitos de alto nível. Além da rastreabilidade com os requisitos de alto nível, também foi implementado um processo para rastrear os requisitos de sistema com os testes de caixa preta.

Para um controle mais granular, foi realizada a rastreabilidade dos requisitos de alto nível com os métodos de caixa branca, para verificar se cada componente funciona conforme o esperado. Já para garantir a rastreabilidade entre os requisitos de alto nível e o código, foi implementada uma estratégia na qual cada identificador único do requisito foi documen-

tado no código fonte, além de existir uma tabela onde cada identificador de requisito está conectado aos nomes de suas principais classes e métodos.

Toda essa rastreabilidade é detalhadamente apresentada nas tabelas do APÊNDICE D.

4.7. Testes

Para a avaliação do software, foram adotados dois métodos de teste: o de caixa branca, utilizando o *framework* gtest, e o de caixa preta, elaborado com base nas possibilidades de interações do usuário.

Os testes de caixa branca foram desenvolvidos com base no código fonte da função a ser testada. Isso nos permite validar a execução correta da função, assegurando que sua saída corresponda às expectativas estabelecidas. A tabela contendo o mapeamento detalhado dos testes de caixa branca está disponível no APÊNDICE F.

Por sua vez, os testes de caixa preta foram concebidos para avaliar o sistema como um todo, considerando uma variedade de cenários de entrada e saída, garantindo que o software atenda aos requisitos do sistema. Os testes de caixa preta são apresentados no APÊNDICE E.

4.8. Benchmarks

Para avaliar a eficácia do método bioinspirado, foram realizados diversos *benchmarks* em comparação com os métodos do IPET e dinâmico. Foram analisados os resultados obtidos do tempo e ciclos de execução de cada método para calcular as diferenças em relação ao método bioinspirado. Para realizar esse *benchmarks* iremos utilizar diversos algoritmos como, por exemplo, o *bubble sort* que possui complexidade O(n^2) em seu pior caso, o *forif* que faz uso de loops de repetição com estruturas condicionais, o *lotfunc* que executa chamadas sucessivas de funções e o *matmul* que faz uso de vários *loops* de repetição aninhados para multiplicação de matrizes.

4.9. Ferramentas Computacionais

No decorrer do desenvolvimento, foram utilizadas diversas ferramentas, agrupadas para atender diferentes aspectos do processo. No conjunto de ferramentas de desenvolvimento, foi empregado Git e Docker para controle de versão e ambientes isolados, respectivamente. Para a construção e compilação do código, foi utilizado Python, Make, GCC (GNU Compiler Collection) e GDB (GNU Debugger). Na esfera de ferramentas de verificação, gTest (Google Test) foi adotado para realizar testes de maneira eficiente. Já as ferramentas incorporadas ao produto abrangem OTAWA, mkff (Make Flow Facts), oRange, OpenOCD (Open On-Chip Debugger), stlink-tools, além de Python e Make.

OTAWA é um framework desenvolvido pelo grupo TRACES da Irit Labs pertencente a Universidade de Toulouse, na França. No projeto, o OTAWA desempenha o papel de geração de Grafos de Fluxo de Controle (CFG) e na descoberta do pior caso de tempo de execução do código através da análise estática utilizando o método IPET.

Os plugins do OTAWA, oRange e mkff, desempenham papéis importantes na análise do código. O mkff gera o arquivo *FlowFacts*, que deve ser preenchido com a quantidade

máxima de iterações que cada laço do código deve executar. O oRange automatiza o processo de determinação da quantidade máxima de iterações de cada laço, para laços com condicionais não muito complexas. A combinação dessas duas ferramentas oferece simplificação para o usuário, eliminando a necessidade de se preencher manualmente a quantidade de iterações para todos os laços do código. Os laços que o oRange não consegue identificar o número de iterações, o usuário deve fornecer quando a ferramenta BioWcet solicitar, sendo um processo mais amigável para o usuário.

O *gTest* é um framework de código aberto para a execução de testes unitários desenvolvido pelo Google, por meio destes testes é provida robustez e a funcionalidade do software desenvolvido. A automação que ele proporciona permite economizar tempo e recursos, além de aumentar a confiabilidade do produto final.

O *OpenOCD* é um software livre criado por Dominic Rath como parte de seu projeto de tese na Universidade de Ciências Aplicadas, FH-Augsburg. Ele é projetado para fornecer suporte de depuração em hardware, permitindo que os desenvolvedores programem, depurem e testem microcontroladores e processadores embarcados. O stlink-tools é um conjunto de ferramentas de software que interagem com dispositivos STM32.Elas podem ser usadas para atividades como programação de firmware, leitura e gravação de memória, depuração e outras operações relacionadas ao desenvolvimento e manutenção de software para microcontroladores STM32. Como parte do ambiente de desenvolvimento, é utilizado também o GCC para a compilar e gerar binários executáveis, bem como o *GDB* para a depuração e análise de código.

Para a automação de tarefas foram utilizados o *make* para a compilação e construção da hierarquia de diretórios assim como o *Python* para automatizar os processos de acesso a registradores do microcontrolador por meio do GDB, assim como a integração entre os dados gerados pelo *oRange* e o *mkff*. Para controle de versionamento de código foi utilizado o *git* e para a disponibilização mais acessível da ferramenta BioWcet foi utilizada uma imagem em *docker*, tornando mais prático o uso da ferramenta.

5. Desenvolvimento

5.1. Versão

A versão 1.0 do software foi o mínimo produto viável (MVP) da implementação dos requisitos básicos, buscando verificar a viabilidade do projeto como um todo. Nessa versão inicial, o software fazia uso do gerador de CFG do OTAWA individualmente, onde cada método de cálculo do WCET instanciava sua própria classe do OTAWA.

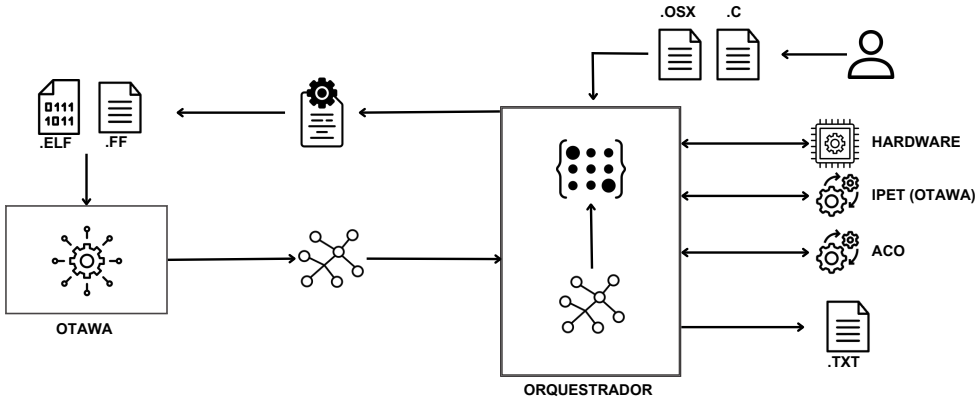
Com a validação bem sucedida do MVP do software, foi desenvolvida a versão 1.10, onde foi realizada a refatoração do código para garantir a conformidade com os requisitos levantados. Para isso, foi criada a classe *cfgGen*, que ficou responsável por criar o *workspace* de trabalho do OTAWA, gerar o CFG do algoritmo, transformar o CFG em uma matriz e calcular o tempo de execução de um bloco básico. Além disso, foram adicionadas informações de porcentagem de andamento do cálculo e uma descrição do nome do software quando o mesmo é iniciado. O presente artigo trata da implementação da versão 1.10 do software.

5.2. Orquestrador e OTAWA

Para implementar a arquitetura idealizada, foi atribuída ao bloco Orquestrador a responsabilidade de manter a conexão entre todos os módulos e centralizar o fluxo de dados de WCET.

Quando o usuário fornece a entrada à ferramenta, ela executa uma preparação dos diretórios que serão utilizados, efetuando a remoção, cópia e renomeação de alguns arquivos. O orquestrador então invoca o método responsável por compilar o arquivo de entrada, tanto para o OTAWA quanto para o microcontrolador, e o método que gera o arquivo *flowfact* e o arquivo *XML*, que possui as estimativas de iterações de laços feitas pelo *oRange*. Na sequência, ele executa a junção dos dois arquivos, o *.ff* e o *XML*. Caso necessário, o orquestrador questiona o usuário sobre as iterações de laços que não foram possíveis de serem estimadas automaticamente.

Figura 4. Diagrama de fluxo do orquestrador.



Na figura 4 é possível visualizar o arquivo *.elf*, gerado para o OTAWA a partir da compilação do arquivo .c, o arquivo .osx, que descreve a arquitetura do microcontrolador analisado fornecido pelo usuário, juntamente com o arquivo de *flowfacts* devidamente preenchido com as iterações de laços estimados. Estes arquivos são carregados em um objeto chamado *workspace*, disponibilizado pelo OTAWA. A fusão de todas essas informações e uma posterior simulação por software da arquitetura, executando o algoritmo alvo, gera o CFG anotado. Para obter o resultado WCET utilizando o método IPET, basta passar o objeto *workspace* para a classe *WCETCalculator*, onde possui métodos, que envolvem o OTAWA, necessários para realizar o cálculo utilizando a técnica mencionada.

Para a análise utilizando o algoritmo ACO, é necessário fazer a conversão do grafo, que foi gerado pelo OTAWA, para uma matriz. O método responsável por essa conversão é o *cfg2Matrix*, pertencente à classe *cfgGen*. Neste método, o workspace, anteriormente citado, é recebido e é utilizado para extrair as características de ciclos consumidos por blocos básicos e a quantidade de passagens por cada bloco. Esta representação matricial se dá por meio de uma estrutura de dados básica para representação de grafos, chamada matriz de adjacências. Nesta representação, o número da linha refere-se ao número do bloco de origem e o número da coluna refere-se ao número do bloco de destino. A quantidade de ciclos consumidos pelo bloco destino está na interseção entre essa linha e coluna.

Devido à natureza pouco densa dos grafos CFG, uma representação convencional de uma

matriz consumiria uma quantidade de memória desnecessária. Portanto, a implementação foi fundamentada na implementação de uma matriz esparsa usando listas ligadas sem cabeça. Na implementação foi decidido que arestas que representam retornos no grafo (que são as arestas referentes a laços) seriam armazenadas em uma matriz distinta das arestas que representam o fluxo da árvore de execução do programa. Para facilitar, uma classe *cfgMatrix* foi criada com a finalidade de abstrair esta representação e facilitar a manipulação de ambas matrizes. Esta decisão de projeto foi tomada para facilitar o processamento dos dados dentro do algoritmo ACO. Após a conversão do grafo na matriz, o algoritmo ACO é executado sobre a *cfgMatrix* fornecendo, então, o valor do WCET.

Na análise dinâmica, o arquivo binário compilado para o microcontrolador é gravado no hardware embarcado. O algoritmo é então executado no hardware e, por meio do GDB, são capturados os tempos de execução. Esse processo será abordado em mais detalhes em 5.4.

5.3. ACO Adaptado

5.3.1. Formulação do Problema de otimização

Para a realizar a análise estática utilizando o ACO, foram realizadas algumas modificações no algoritmo, visto que ele foi projetado inicialmente para encontrar o menor caminho, enquanto o objetivo deste trabalho é encontrar o oposto. Assim, a quantidade de feromônio depositado no caminho de i para j será de L_k.

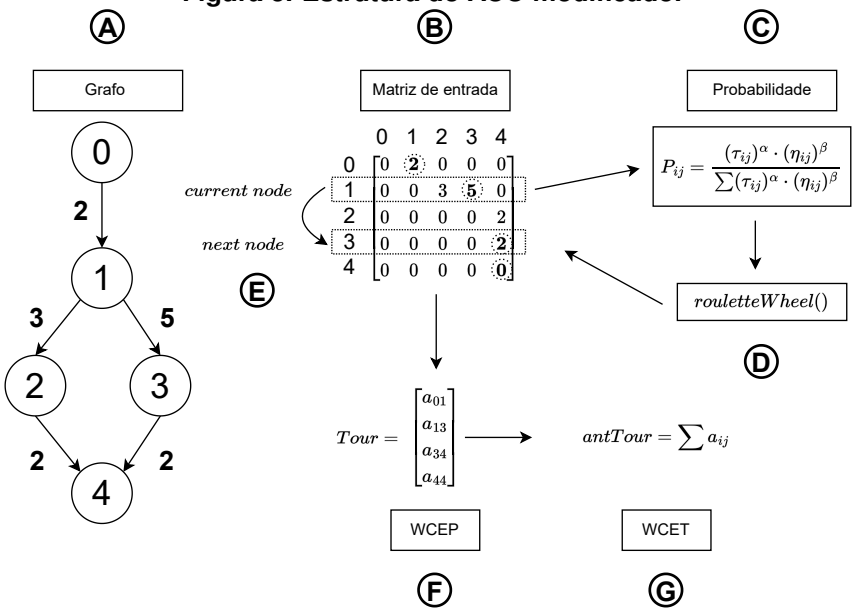
A Figura 5 mostra o processo de obtenção do WCEP e WCET para grafos condicionais (if-else e case) e grafos sequenciais. O orquestrador, utilizando o OTAWA, realiza a análise de fluxo de controle dos blocos básicos e a obtenção do tempo de execução individualmente, resultando no grafo do programa em **A**. Em seguida, o grafo é tratado para a obtenção de uma matriz de adjacências, onde as linhas representam os possíveis caminhos a partir de um determinado nó em **B**. À medida em que as formigas andam pelo grafo, em cada nó é realizado um cálculo de probabilidade de todos os possíveis caminhos, como mostrado em **C**, e então é aplicado o método da roleta em **D**. Em sequência, é verificado o próximo nó, no qual será realizado o mesmo procedimento até chegar no fim da matriz, como mostrado em **E**. O caminho percorrido pela formiga é armazenado em um vetor *tour*, que representa o WCEP daquela iteração, como pode ser visualizado em **F**. Por fim, é realizada a soma de todos os elementos do vetor correspondente ao WCET encontrado pela formiga, conforme apresentado em **G**.

Em seguida, é feita uma comparação entre os resultados encontrados pelas formigas daquela iteração, selecionando a melhor entre elas, a que tem o maior caminho (a rainha). O WCEP e WCET são armazenados na rainha (*queenTour* e *queenAntTour*). A função *fitness* é definida por:

$$fitness = max(queenAntTour). \tag{6}$$

Além disso, a taxa de feromônio também é atualizada a cada iteração com base na rainha encontrada.

Figura 5. Estrutura do ACO modificado.



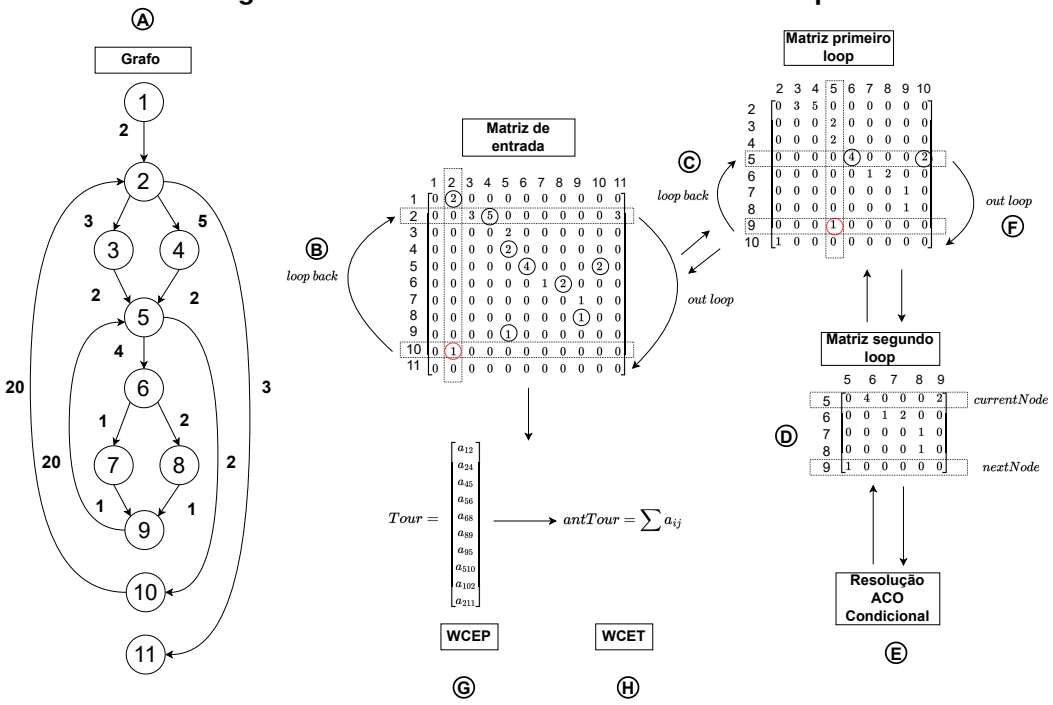
5.3.2. Otimização com loops

Realizar o cálculo do WCET utilizando o ACO em loops é um desafio, pois nem sempre o grafo tem os blocos básicos em ordem numérica sequencial. Assim, foi projetado um método para encontrar o começo e o final do loop, no qual se verifica tanto a linha quanto a coluna do nó atual. Na matriz, a linha representa para quais nós a formiga pode se mover, e a coluna mostra se há algum nó que faz a formiga retornar ao lugar inicial (*loop back*). Com isso, é possível determinar onde começa e onde termina um loop. Desse modo, a formiga é colocada para percorrer o loop a quantidade de vezes que foi determinada pelo usuário.

A Figura 6 mostra o processo de cálculo do WCET e WCEP para grafos com loop. Em **A**, assim como no processo anterior, o grafo é tratado e transformado em uma matriz, conforme apresentada em **B**. Cada vez que a formiga anda na matriz, é verificado se há um loop. Caso seja encontrado, a formiga começa a andar entre o início e o fim do loop, como mostrado em **C**. Esse processo de encontrar loops continua recursivamente até encontrar uma matriz de grafo condicional, como apresentado em **D**. O processo de resolução segue o mesmo padrão descrito em 5.3.1 e apresentado em **E**, onde a formiga fica andando no loop até que complete a quantidade máxima especificada. Ao terminar o loop, a formiga é forçada a sair através do (*out loop*), conforme apresentado em **F**. Após resolver todos os loops, o caminho é armazenado no vetor (*Tour*), e a soma dos elementos do vetor é registrada em (*antTour*), conforme destacado em **G** e **H**.

O processo para obter os melhores valores do WCET e WCEP segue a mesma abordagem da matriz condicional.

Figura 6. Processo de cálculo do ACO com Loops.



5.3.3. Solução de máximos locais

Para mitigar o problema dos máximos locais no ACO durante a busca por soluções ótimas, estratégias específicas podem ser implementadas. Uma abordagem eficaz é fazer com que a formiga enxergue o caminho além do próximo nó, sendo o valor adotado de 2 nós à frente, esse número foi escolhido devido a probabilidade de 2 blocos básicos com consumos de ciclos pequenos seguidos ser menos provável. Essa técnica aumenta assim o nosso potencial de desviar de máximos locais, evitando convergências prematuras para soluções sub ótimas, aumentando as chances de encontrar soluções globalmente ótimas.

5.4. Estimativa de tempo de execução dinâmica

O cálculo do tempo de execução do algoritmo por meio do hardware é desafiador, pois a instrumentação utilizada para realizar a medida deve afetar o mínimo possível o valor da medida e, levando-se isso em consideração, a plataforma desenvolvida utiliza poucos comandos para fazer essa amostragem do ponto de vista do dispositivo embarcado.

Na arquitetura ARM, especificamente na família de processadores M, temos um registrador de fácil acesso que contabiliza a quantidade de ciclos de clock. Este registrador, uma vez configurado com o *prescaler* correto, é identificado pela macro *ARM_CM_DWT_CYCCNT* dentro do driver de instrumentação implementado, chamado de *cron*.

No driver de instrumentação, temos as funções *setupCronometer*, que inicializa corretamente o registrador para a contagem de ciclos, *startCronometer*, que guarda o valor naquele instante registrado pelo contador e a função *stopCronometer*, que retorna a diferença entre a quantidade de ciclos registrada pelo *startCronometer* e no momento em que

stopCronometer foi chamado.

As chamadas a esta biblioteca devem ser restritas apenas à medida dinâmica e não incluídas na medida estática, para evitar interferências. Portanto, as chamadas desta função devem ser protegidas por uma flag de compilação. Desta forma, para obter os arquivos binários necessários para análise estática do WCET, é feita a compilação sem a flag *CRON*, e para a análise dinâmica, a flag de compilação é habilitada. No código 1, podemos ver um exemplo de utilização do driver de instrumentação implementado.

Código 1. Exemplo de uso da biblioteca cron

```
#ifdef CRON
    #include "cron.h"
#endif

int main () {
#ifdef CRON
    setupCronometer();
    uint32_t start = startCronometer();
#endif
    int a = 1;
    if (a == 1) {
        return a;
    }
#ifdef CRON
    uint32_t cycles = stopCronometer(start);
#endif
    return 0;
}
```

Após a geração do arquivo binário instrumentado, é feita a programação do microcontrolador com este arquivo utilizando a ferramenta *stlink-tools*. Com o microcontrolador com sua memória *flash* programada, é feita a inicialização da ferramenta do *OpenOCD*, que, a partir dos dados de endereçamento do processador, cria uma interface de servidor com o microcontrolador. Vale ressaltar que é necessário executar o *OpenOCD* em uma *thread* separada, visto que a comunicação com a *thread* que executa a ferramenta ficará pendurada até que a mesma seja forçada a parar.

Feita a inicialização do *OpenOCD*, é feita a execução da ferramenta do GDB, utilizando o *script hwdebug.py* e o arquivo binário gerado como entradas. O *script* de *debug* irá definir como alvo o servidor criado pelo *OpenOCD*, para que seja feita a comunicação com o microcontrolador. Após realizado o acesso ao microcontrolador, o *script* reinicia o processador, cria um ponto de parada após a medição do tempo de execução e deixa o processador seguir até que seja feita a parada.

Após a parada, o valor de ciclos desta iteração é armazenado em um vetor. O processo de reinicialização e medição é feito em loop, que é executado um total de 10 vezes. O valor estimado do WCET dinâmico é então o maior valor encontrado no vetor onde possui cada medição no processador.

Para estimar o número de ciclos de *clock* que o processo de instrumentação do código estava gastando em cada medição, foi feito o mesmo processo de cronometragem, só que desta vez utilizando uma instrução de NOP como referência. Desta forma, ao obter o valor do WCET dinâmico deste código, é realizado o desconto de um ciclo referente a instrução NOP e é obtido a variação da medição referente a instrumentação do código.

6. Resultados

6.1. Setup

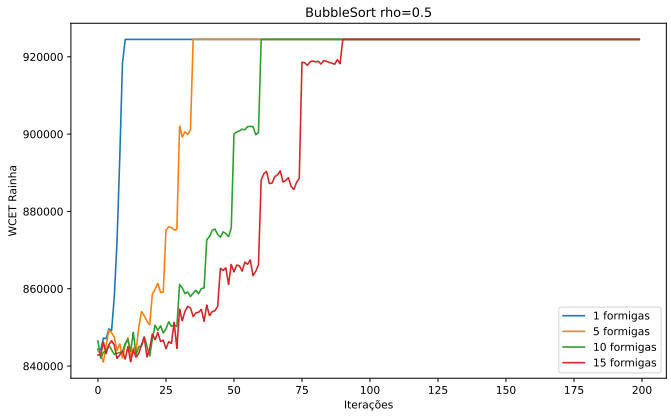
A configuração paramétrica para o ACO depende do problema de otimização. No projeto, foi utilizada uma população de 10 formigas, mantendo os parâmetros α e β iguais a 1. Isso foi realizado para que a probabilidade de uma formiga escolher um caminho fosse mais justa, com chances iguais para cada caminho, dependendo do peso da aresta. A taxa de vaporização ρ é definida como 50 % para diminuir a possibilidade de convergências prematuras e evitar o consumo excessivo de recursos computacionais. Os testes foram realizados em um notebook com processador Intel Core-i7 10510U de 10° geração, operando a uma frequência base de 1.8 GHz $\times$ 8, com 16 GB de memória RAM e sistema operacional Ubuntu 22.04 LTS de 64bits.

6.2. Resultados ACO

6.2.1. Análise de Convergência

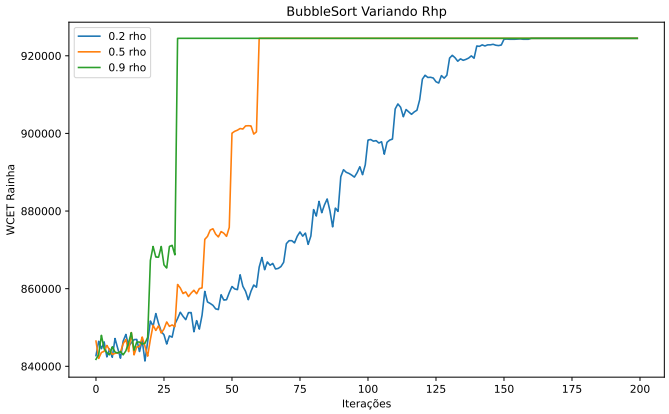
A Figura 7 mostra a curva de convergência do ACO variando a quantidade de formigas entre 1 e 15 formigas com 200 iterações para o algoritmo *bubble sort* sobre um vetor de 100 posições contemplando o pior caso. Este algoritmo foi escolhido por ter loops aninhados e ser um dos casos dos testes do *benchmark* proposto. O ACO com 1 formiga apresenta uma convergência mais rápida, porém com 15 formigas há uma maior exploração de caminhos possíveis.

Figura 7. Comparativo da curva de convergência para cálculo do WCET para o algoritmo bubble sort de 100 posições utilizando o ACO e variando a quantidade de formigas.



A Figura 8 mostra o impacto da taxa de vaporização ρ na trilha deixada pelas formigas e também na convergência do WCET, onde a taxa de $\rho = 0.2$ teve uma retenção de informações maior entre as iterações já que o feromônio permanece mais tempo na trilha permitindo maiores explorações de caminhos e $\rho = 0.9$ a vaporização é mais rápida e pode resultar na exploração de trilhas específicas que foram exploradas com mais sucesso, resultando em convergências mais rápidas.

Figura 8. Comparativo da curva de convergência para cálculo do WCET para o algoritmo bubble sort de 100 posições utilizando o ACO, fixando o número de formigas em 10 e variando a taxa de vaporização do feromônio.



6.2.2. Análise de Grafo

A Figura 9 mostra o número de passagens das formigas pelo grafo sem o método previsor e a Figura 10 mostra o mesmo cenário com o método previsor para o código *forif*. O método previsor fez com que as formigas explorassem mais o grafo encontrando melhores caminhos e evitando máximos locais.

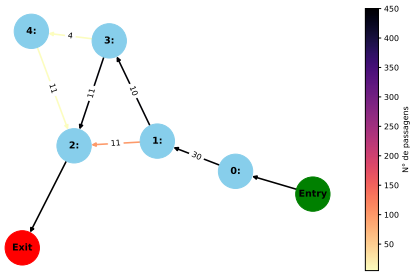


Figura 9. ACO sem método previsor

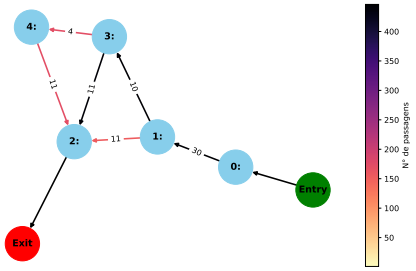


Figura 10. ACO com método previsor

6.3. Resultados Integração

Neste capítulo, serão apresentados os resultados obtidos após a integração das 3 metodologias de cálculo do WCET disponíveis na ferramenta. Ao utilizar algoritmos seleciona-

dos para testar a ferramenta, foi possível obter os resultados apresentados nas Tabelas 2 e 3. Com os resultados obtidos, foi feita uma análise de desempenho das metodologias e uma comparação do resultado das mesmas.

Tabela 2. Resultados de WCET obtidos pela ferramenta BioWCET utilizando os algoritmos propostos no benchmark.

Algoritmo	WCET IPET [ciclos]	WCET ACO [ciclos]	WCET DIN [ciclos]	Erro IPET-ACO	Erro DIN-ACO
nop	17	17	1	0,00%	1600,00%
neasted	1039	1063	671	2,31%	58,42%
matmul	103513	104511	72588	0,96%	43,98%
lotfunc	582	582	397	0,00%	46,60%
if	55	55	22	0,00%	150,00%
func	2084	2160	951	3,65%	127,13%
forif	381	399	241	4,72%	65,56%
fir	25984	26816	16327	3,20%	64,24%
bsort100	907964	927662	345782	2,17%	168,28%
bs	3280	3328	2313	1,46%	43,88%

A Tabela 2 apresenta os valores de WCET obtidos após executar a ferramenta com o código de cada um dos algoritmos selecionados. O WCET IPET representa o número de ciclos obtidos utilizando o método IPET, o WCET ACO representa o número de ciclos obtidos utilizando o ACO e o WCET DIN representa o número de ciclos obtidos a partir da medição no hardware proposto. O erro IPET-ACO representa o erro percentual absoluto obtido através da comparação entre o valor do WCET ACO relativo ao WCET IPET. O erro DIN-ACO representa o erro percentual absoluto obtido através da comparação entre o valor do WCET ACO relativo ao WCET DIN. O algoritmo IPET é utilizado como referência por já ser reconhecido como referência para este cálculo.

Ao analisarmos os valores de erro em relação ao IPET, pode-se ver que os valores são muito próximos, tendo um erro máximo encontrado de 4.72% nos casos de teste selecionados. Já observando o erro relacionado ao método dinâmico, pode-se ver que existe uma disparidade muito grande entre os valores. Esta disparidade sempre entrega uma estimativa pessimista em relação ao tempo calculado em hardware, o que era de se esperar de uma medida de WCET correta.

A Tabela 3 mostra os tempos de execução de cada um dos algoritmos utilizados para calcular o WCET. Foi levado em consideração o tempo de execução do IPET, o tempo de execução do algoritmo do ACO considerando os casos em que foram utilizadas 1, 2 e 5 formigas e por fim o tempo de medição dinâmica em hardware. Estes tempos foram calculados para cada um dos algoritmos do *benchmark* proposto.

Ao observarmos os tempos de execução do algoritmo bioinspirado com diferentes números de formigas, pode-se ver que ao aumentarmos a quantidade de formigas no ACO, o tempo de execução do algoritmo aumenta consideravelmente, porém não existe uma linearidade entre o número de formigas e o tempo de execução.

Pode-se observar também que o tempo de execução do algoritmo do ACO aumenta de acordo com a complexidade do algoritmo analisado. Ao calcularmos o coeficiente de variação para cada um dos tempos de execução com diferente quantidade de formigas, encontra-se que para 1 formiga, tem-se o coeficiente de 268,69%; para 5 formigas, tem-se o coeficiente de 293,57% e por fim, para 10 formigas, tem-se o coeficiente de 331,50%. Estes dados mostram que o aumento no número de formigas no ACO aumenta a variabi-

Tabela 3. Resultados de tempo de execução obtidos pela ferramenta BioWCET utilizando os algoritmos proposto no benchmark.

Algoritmo	TE IPET [us]	TE ACO 1 formiga [ms]	TE ACO 5 formigas [ms]	TE ACO 10 formigas [ms]	TE DIN [ms]
nop	0,216	0,537071	3,557445	10,642185	2462,265984
neasted	0,217	22,515877	38,74187	67,449712	2459,638524
matmul	0,218	65,340481	98,463738	155,885375	2664,491786
lotfunc	0,22	4,103702	6,968743	16,609335	2580,046435
if	0,221	1,162707	3,575731	7,405321	2551,464363
func	0,198	3,24911	3,921263	9,002352	2451,468063
forif	0,214	8,552715	12,061384	20,481436	2486,106689
fir	0,211	189,06129	289,672247	500,912061	3139,400724
bsort100	0,193	2255,384405	2982,662932	5009,479675	2520,747047
bs	0,192	7,035532	8,93425	22,3972	2608,775859

lidade do tempo de execução dos algoritmos selecionados.

Analisando o tempo de execução do IPET, é possível calcular o coeficiente de variação das amostras. Desta forma, o coeficiente de variação foi de 12.76%, o que mostra que os tempos de execução não variam com a complexidade do algoritmo de entrada. Fazendo o mesmo cálculo para o tempo de execução dinâmico, o coeficiente de variação foi de 3.03%, mostrando que o tempo de execução também varia pouco para cada algoritmo.

Comparando o tempo de execução do IPET com o tempo de execução do algoritmo bioinspirado com 10 formigas, temos que o IPET é mais rápido que o ACO e não tem uma variabilidade relacionada a complexidade do algoritmo proposto. Ao comparar o tempo de execução do método baseado em medições com o bioinspirado com 10 formigas, pode-se ver que apenas um algoritmo teve um tempo de execução maior com o bioinspirado.

6.4. Resultados dos testes

Nos testes de caixa preta, primeiramente, identificaram-se as possíveis variáveis de entrada: Arquitetura, Arquivo e Flag. A partir disso, criaram-se suas classes de equivalência e foi identificado que as variáveis Arquivo e Arquitetura poderiam formar pares ortogonais. Com a classe de equivalência da variável Flag e os pares ortogonais das variáveis Arquivo e Arquitetura, elaborou-se a matriz de testes possíveis. A partir dessa matriz, identificaram-se 24 casos de teste, e, devido à pequena quantidade, optou-se pela abordagem de teste robusta forte, onde ocorre a combinação de todos os valores de forma exaustiva. Os resultados desses testes estão detalhados no APÊNDICE H .

Para o teste de caixa branca, foi efetuado o rastreamento dos requisitos com suas respectivas funções públicas testáveis no software. Posteriormente, elaborou-se um caso de teste para cada função, verificando se o resultado era compatível com requisito estabelecido. O *gtest* trabalha com informações de *ok* e *fail* para informar se o teste foi executado com sucesso ou não. Foram escritos 11 testes de unidade das funções públicas e, como poder ser verificado no APÊNDICE G, todos os testes foram executados com êxito, ficando evidenciado que o software estava funcionando de acordo com os seus requisitos.

7. Conclusão

O cálculo do WCET é uma métrica essencial para garantir que sistemas de tempo real atendem aos seus requisitos de tempo. A ferramenta implementada utilizou 3 formas

diferentes de cálculo de WCET: uma análise estática utilizando o IPET, uma análise dinâmica medida no hardware proposto e outra análise estática utilizando o algoritmo bioinspirado ACO, sendo o último um método inovador pra esse objetivo, visto que, até onde é conhecido, é a primeira vez que ele está sendo utilizado pra este propósito.

Os comparativos com o algoritmo do IPET implementado pela ferramenta OTAWA mostram que os valores obtidos pelos dois algoritmos são bem próximos. Ao compararmos com o método baseado em medições ocorreu uma variação maior dos valores. Esta variação se dá devido ao modelo de hardware implementado para a ferramenta OTAWA. A partir deste modelo de hardware, a ferramenta gera um grafo com as anotações de tempo de cada bloco, sendo este grafo utilizado para o cálculo do IPET pela própria ferramenta e pelo BioWCET como entrada do ACO. Desta forma, se o modelo estiver incorreto, as estimativas de WCET estáticas não terão precisão para o hardware proposto.

O algoritmo do ACO apresentou um tempo de execução maior que os demais métodos devido a estrutura de buscas nos loops aninhados e também ao fato do algoritmo fazer uso de matrizes, o que pode ser computacionalmente complexo. Porém, é possível reduzir este tempo realizando otimizações no código. Outra forma de reduzir este tempo é modificar a forma de percorrer loop, visto que na forma implementada, toda formiga deve passar pelo mesmo loop o número de vezes que ele pode ser executado. Além disso, é possível também executar o algoritmo do ACO em uma GPU, visto que ela pode acelerar os cálculos matriciais. Apesar das variações, os resultados do ACO apresentam um potencial promissor para a estimativa do WCET, indicando sua viabilidade como alternativa.

O algoritmo bioinspirado do ACO mostrou uma grande capacidade para lidar com problemas complexos e não lineares, como o cálculo do WCET, além de conseguir explorar os diferentes caminhos do grafo de entrada enquanto refina a busca pelo pior caminho utilizando os feromônios. O uso de ACO para estimar o WCET em sistemas embarcados revelou-se uma abordagem inovadora e promissora e, até onde é conhecido, esse é o primeiro estudo que utiliza esse algoritmo para cálculo do WCET.

8. Trabalhos futuros

Para trabalhos futuros no projeto, é visto a possibilidade de implementação de novas arquiteturas de hardware para teste a partir da modularização do hardware. A partir de modelos de hardware criados para a ferramenta OTAWA, é possível gerar o WCET utilizando os algoritmos do IPET e do ACO. Além disso, seria necessário implementar a medição do tempo de execução para o hardware escolhido.

Outro trabalho futuro importante para o projeto é a implementação de um modo mais rápido de cálculo de WCET para loops utilizando o ACO. Como nesta versão as formigas passam pelos loops o número máximo de vezes que o loop itera, é proposto que, para reduzir o custo computacional, as formigas executem apenas uma vez o loop e o valor de WCET encontrado nesta única iteração seja multiplicado pelo número de vezes que o loop pode ser executado. Desta forma, o tempo de execução do algoritmo do ACO sofrerá uma redução para algoritmos mais complexos.

Visando que futuramente essa ferramenta poderá ser refinada e certificada, se faz necessário como trabalho futuro um estudo probabilístico com testes de hipótese de modo que

para códigos com menores impactos poderia ser considerado um intervalo de confiança de 95 % e para análise de WCET de códigos criticos um intervalo de confiança de 99 %.

Como melhorias vistas para o que já foi desenvolvido no trabalho, tem-se a melhor adaptação do modelo de hardware utilizado na ferramenta OTAWA. A partir desta melhoria, é possível que o WCET obtido de forma dinâmica se aproxime mais do tempo obtido pela análise estática. Outro ponto importante é a melhoria da modularização do código, separando funções dos blocos de interface e do orquestrador do arquivo principal. Por fim, outra possível melhoria é a forma de utilizar a estrutura de dados criada na biblioteca *cfgMatrix* de forma mais eficiente, visto que muitas das funções para tratar vetores que poderiam otimizar o código não foram utilizadas.

Referências

Savi motivation. `https://savi.avsi.aero/about-savi/savi-motivation/`. Acesso em: 28 de março de 2024.

Abella, J., Hernandez, C., Quiñones, E., Cazorla, F. J., Conmy, P. R., Azkarate-askasua, M., Perez, J., Mezzetti, E., and Vardanega, T. (2015). Wcet analysis methods: Pitfalls and challenges on their trustworthiness. In *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–10.

Ballabriga, C., Forget, J., and Lipari, G. (2015). Context-sensitive Parametric WCET Analysis. In *15th International Workshop on Worst-Case Execution Time Analysis*, Lund , Sweden.

Behera, N. (2020). Chapter 8 - analysis of microarray gene expression data using information theory and stochastic algorithm. In Srinivasa Rao, A. S. and Rao, C., editors, *Principles and Methods for Data Science*, volume 43 of *Handbook of Statistics*, pages 349–378. Elsevier.

Benedicte, P., Kosmidis, L., Quiñones, E., Abella, J., and Cazorla, F. J. (2016). A confidence assessment of wcet estimates for software time randomized caches. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 90–97.

Bygde, S., Ermedahl, A., and Lisper, B. (2009). An efficient algorithm for parametric wcet calculation. In *2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 13–21.

de Oliveira, R. S. (2020). Fundamentos dos sistemas de tempo real.

Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, 26(1):29–41.

Ermedahl, A., Stappert, F., and Engblom, J. (2005). Clustered worst-case execution-time calculation. *IEEE Transactions on Computers*, 54(9):1104–1122.

Huber, B. and Schoeberl, M. (2009). Comparison of Implicit Path Enumeration and Model Checking Based WCET Analysis. In Holsti, N., editor, *9th International Workshop*

on Worst-Case Execution Time Analysis (WCET'09), volume 10 of *Open Access Series in Informatics (OASIcs)*, pages 1–12, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Huybrechts, T., Mercelis, S., and Hellinckx, P. (2018). A new hybrid approach on wcet analysis for real-time systems using machine learning. In *18th International Workshop on Worst-Case Execution Time Analysis (WCET 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Jean, X., Girbal, S., Roger, A., Megel, T., and Brindejonc, V. (2015). Safety considerations for wcet evaluation methods in avionic equipment. In *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pages 7A4–1–7A4–15.

Kebbal, D. (2006). Automatic flow analysis using symbolic execution and path enumeration. In *2006 International Conference on Parallel Processing Workshops (ICPPW'06)*, pages 8 pp.–404.

Kumar, V. (2022). An integrated approach of genetic algorithm and machine learning for generation of worst-case data for real-time systems. In *2022 IEEE/ACM 26th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 87–95.

Lee, J., Shin, S. Y., Nejati, S., Briand, L., and Parache, Y. I. (2023). Estimating probabilistic safe wcet ranges of real-time systems at design stages. *ACM Transactions on Software Engineering and Methodology*, 32(2):1–33.

Lv, M., Guan, N., Zhang, Y., Deng, Q., Yu, G., and Zhang, J. (2009). A survey of wcet analysis of real-time operating systems. In *2009 International Conference on Embedded Software and Systems*, pages 65–72.

Silva, K. P. et al. (2015). Análise de valor para determinação do tempo de execução no pior caso (wcet) de tarefas em sistemas de tempo real.

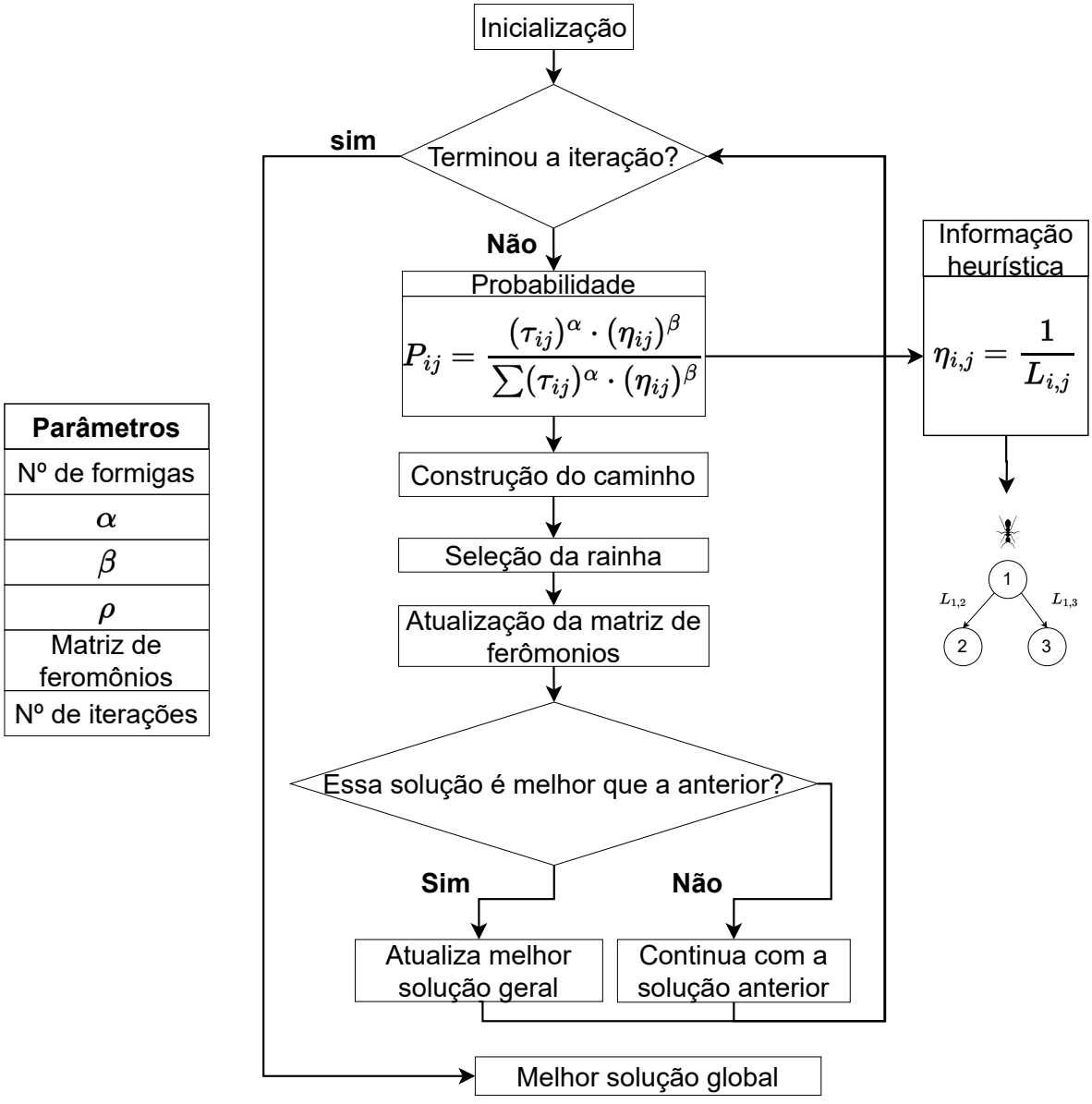
Silva, K. P. et al. (2019). Contributions to the estimation of the worst-case execution time using measurements in real-time systems.

Stützle, T. and Hoos, H. H. (2000). Max–min ant system. *Future generation computer systems*, 16(8):889–914.

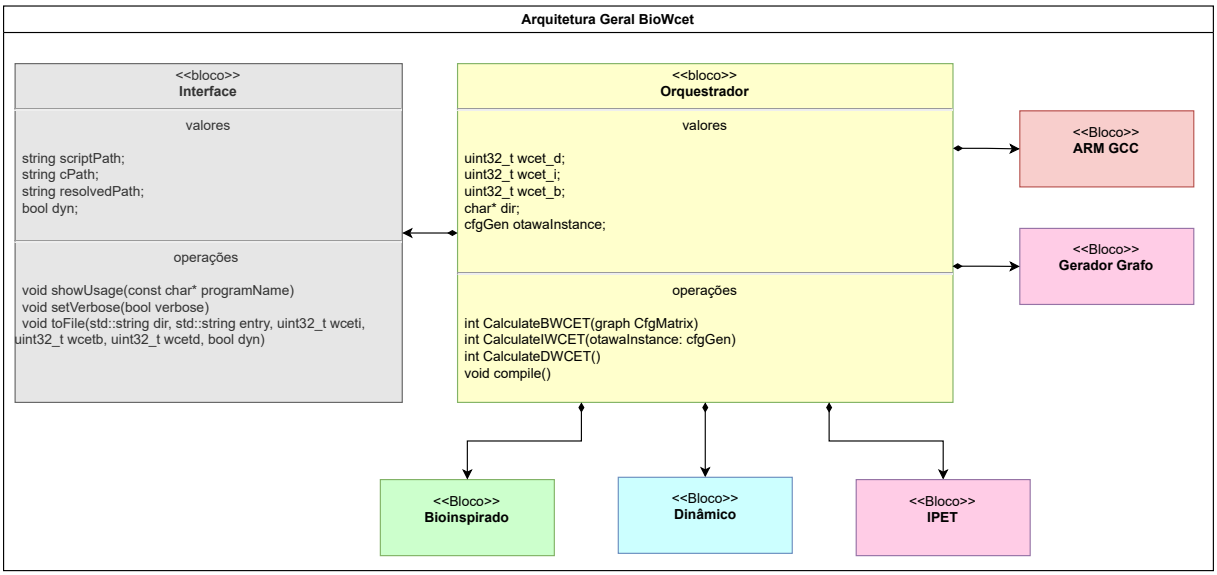
Wegener, J. and Mueller, F. (2001). A comparison of static analysis and evolutionary testing for the verification of timing constraints. *Real-time systems*, 21:241–268.

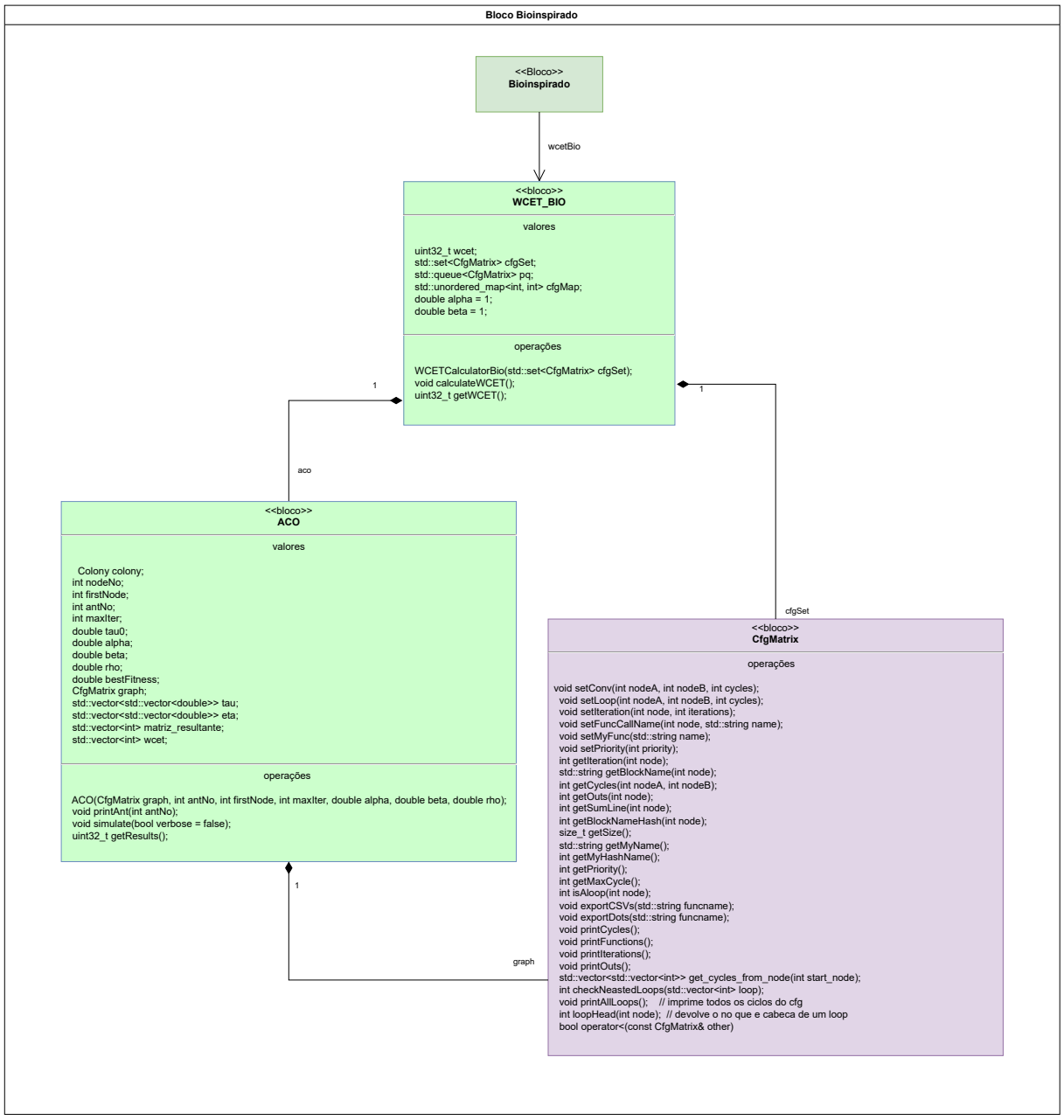
Apêndices

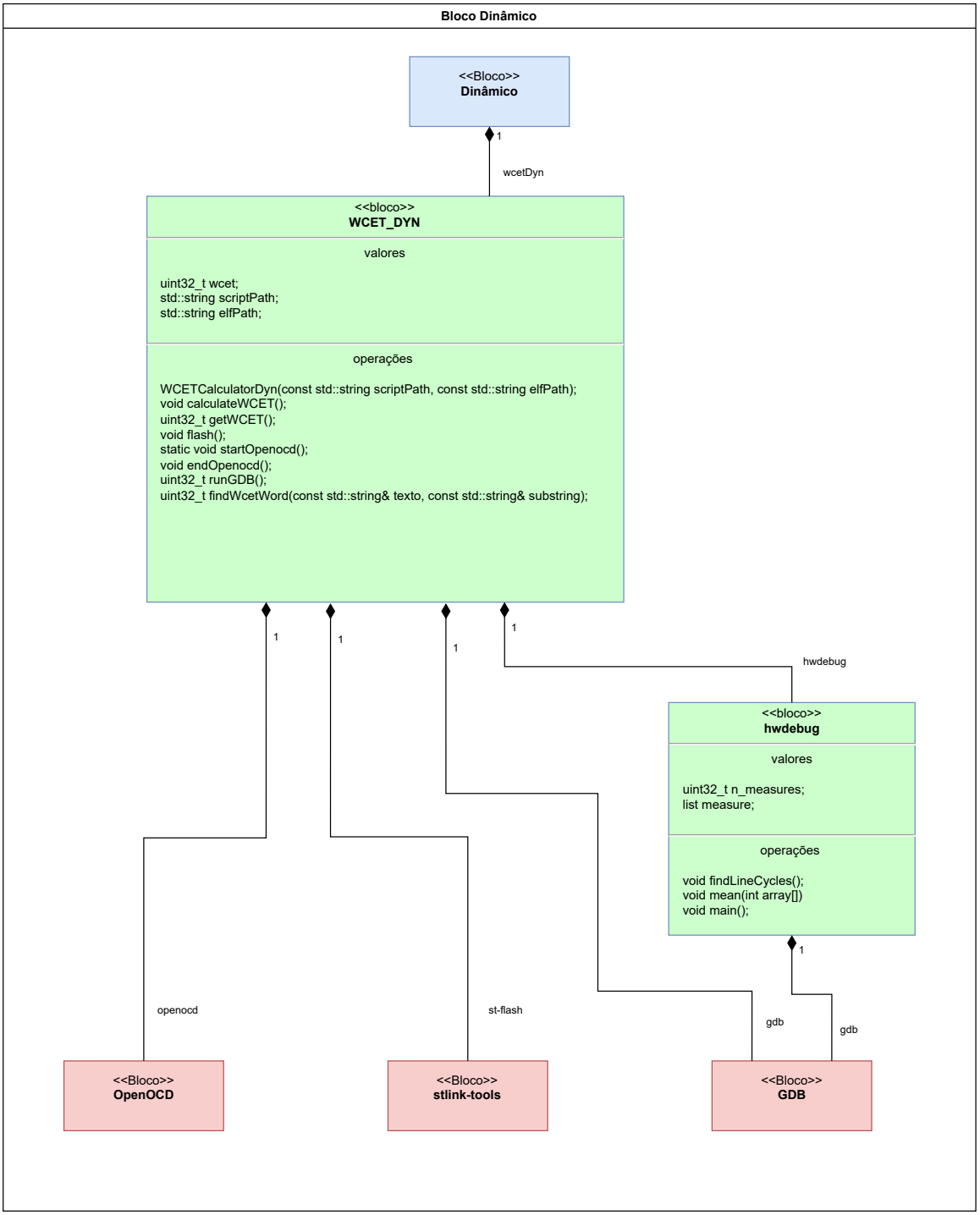
A. Fluxograma ACO

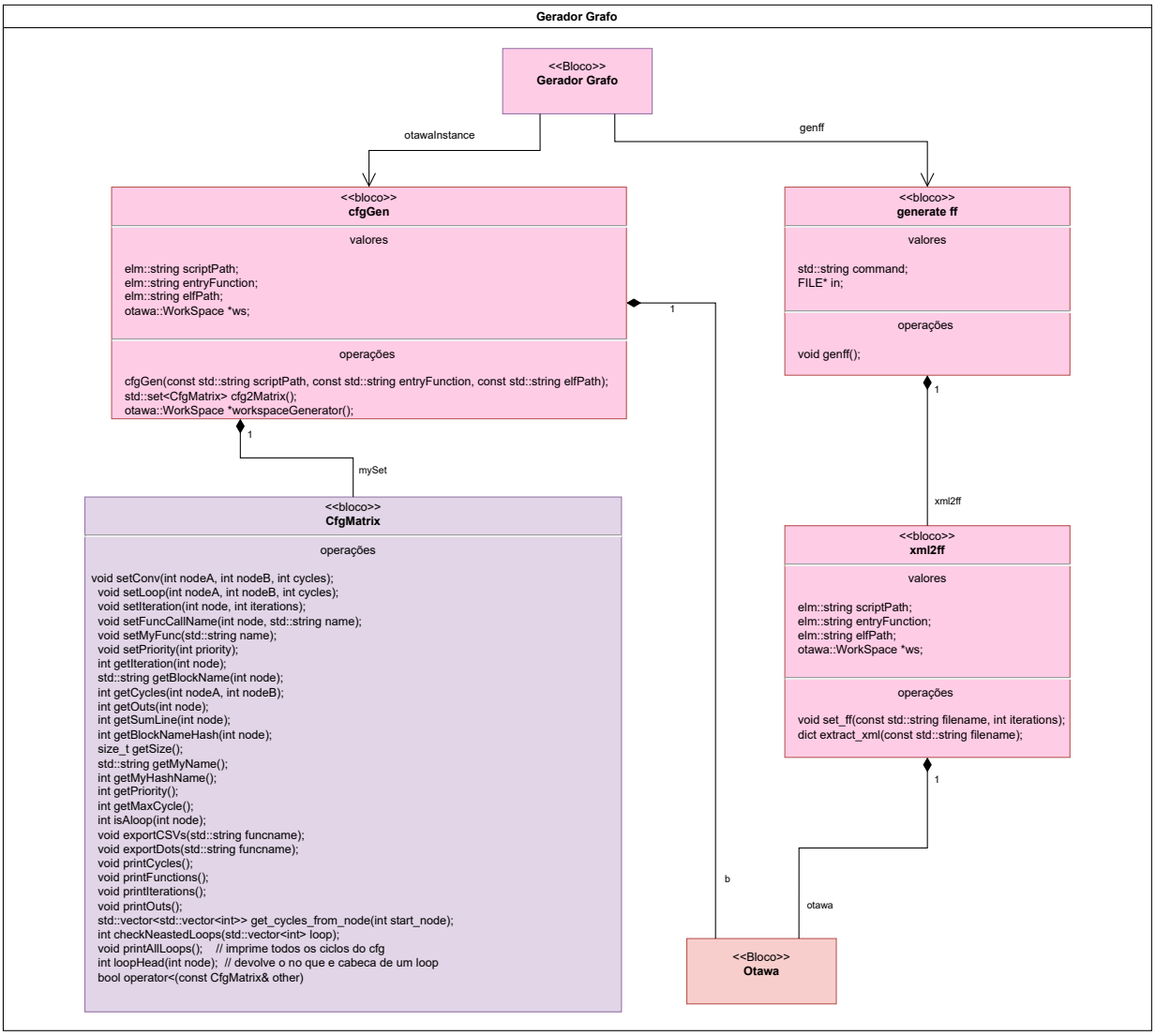


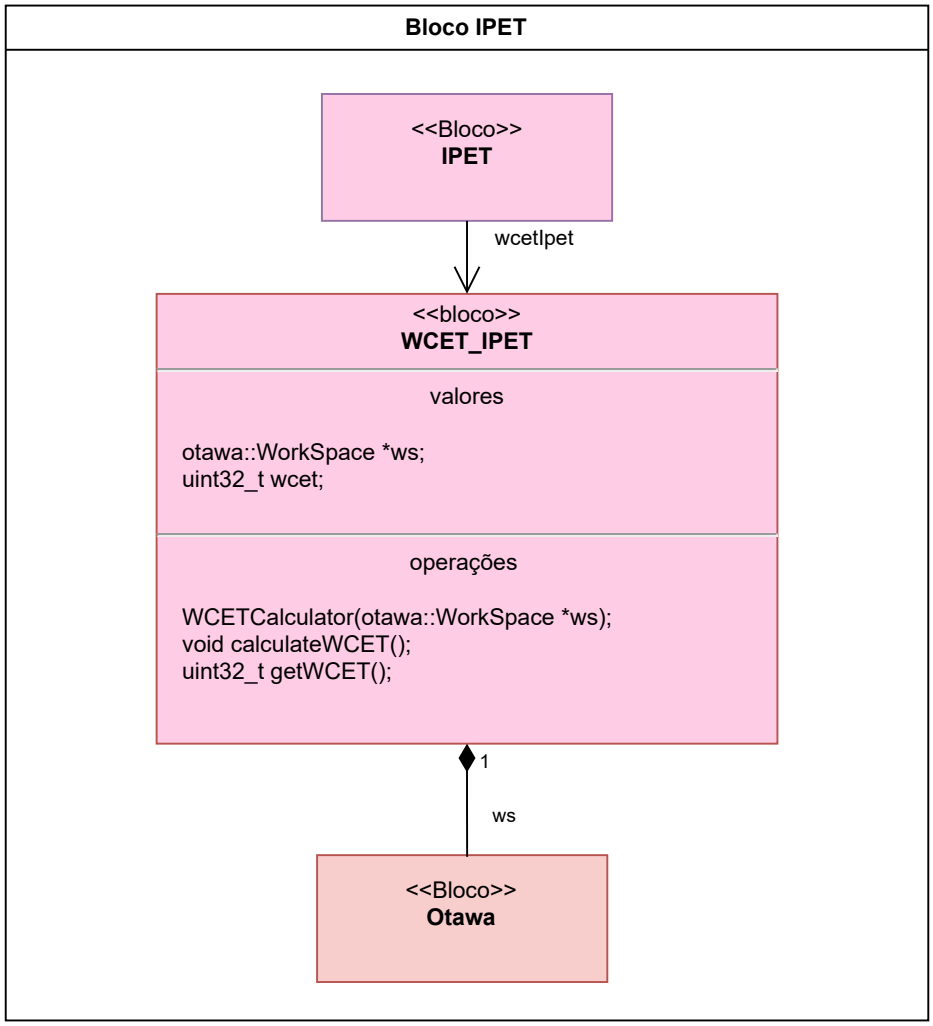
B. Arquitetura

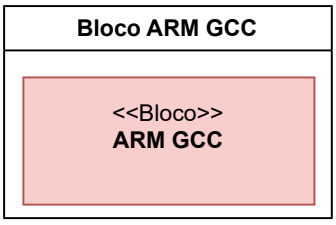


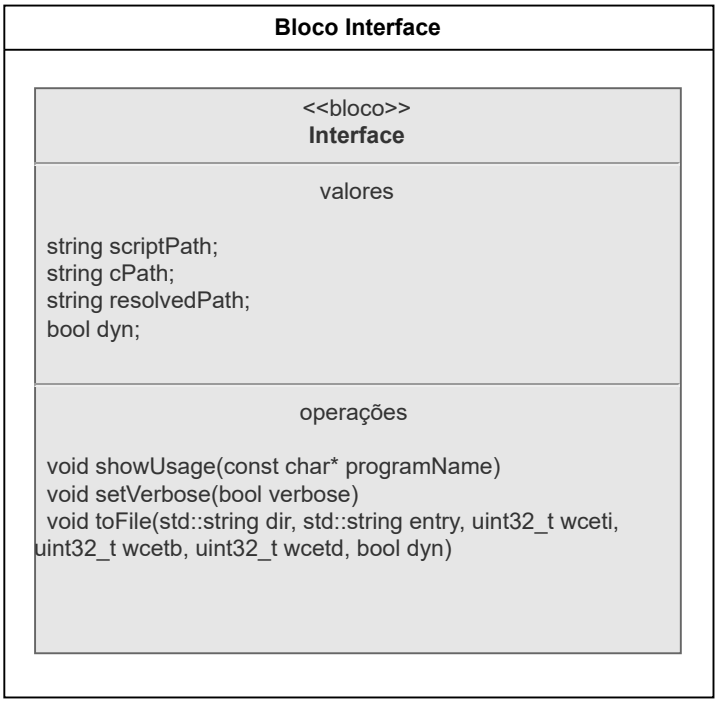

Bloco Bioinspirado
<<Bloco>>
Bioinspirado
wcetBio
<<bloco>>
WCET_BIO
valores
uint32_t wcet;
std::set<CfgMatrix> cfgSet;
std::queue<CfgMatrix> pq;
std::unordered_map<int, int> cfgMap;
double alpha = 1;
double beta = 1;
operações
WCETCalculatorBio(std::set<CfgMatrix> cfgSet);
void calculateWCET();
uint32_t getWCET();
1
1
aco
<<bloco>>
ACO
valores
Colony colony;
int nodeNo;
int firstNode;
int antNo;
int maxIter;
double tau0;
double alpha;
double beta;
double rho;
double bestFitness;
CfgMatrix graph;
std::vector<std::vector<double>> tau;
std::vector<std::vector<double>> eta;
std::vector<int> matriz_resultante;
std::vector<int> wcet;
operações
ACO(CfgMatrix graph, int antNo, int firstNode, int maxIter, double alpha, double beta, double rho);
void printAnt(int antNo);
void simulate(bool verbose = false);
uint32_t getResults();
cfgSet
<<bloco>>
CfgMatrix
operações
void setConv(int nodeA, int nodeB, int cycles);
void setLoop(int nodeA, int nodeB, int cycles);
void setIteration(int node, int iterations);
void setFuncCallName(int node, std::string name);
void setMyFunc(std::string name);
void setPriority(int priority);
int getIteration(int node);
std::string getBlockName(int node);
int getCycles(int nodeA, int nodeB);
int getOuts(int node);
int getSumLine(int node);
int getBlockNameHash(int node);
size_t getSize();
std::string getMyName();
int getMyHashName();
int getPriority();
int getMaxCycle();
int isAloop(int node);
void exportCSVs(std::string funcname);
void exportDots(std::string funcname);
void printCycles();
void printFunctions();
void printIterations();
void printOuts();
std::vector<std::vector<int>> get_cycles_from_node(int start_node);
int checkNeastedLoops(std::vector<int> loop);
void printAllLoops(); // imprime todos os ciclos do cfg
int loopHead(int node); // devolve o no que e cabeca de um loop
bool operator<(const CfgMatrix& other)
1
graph

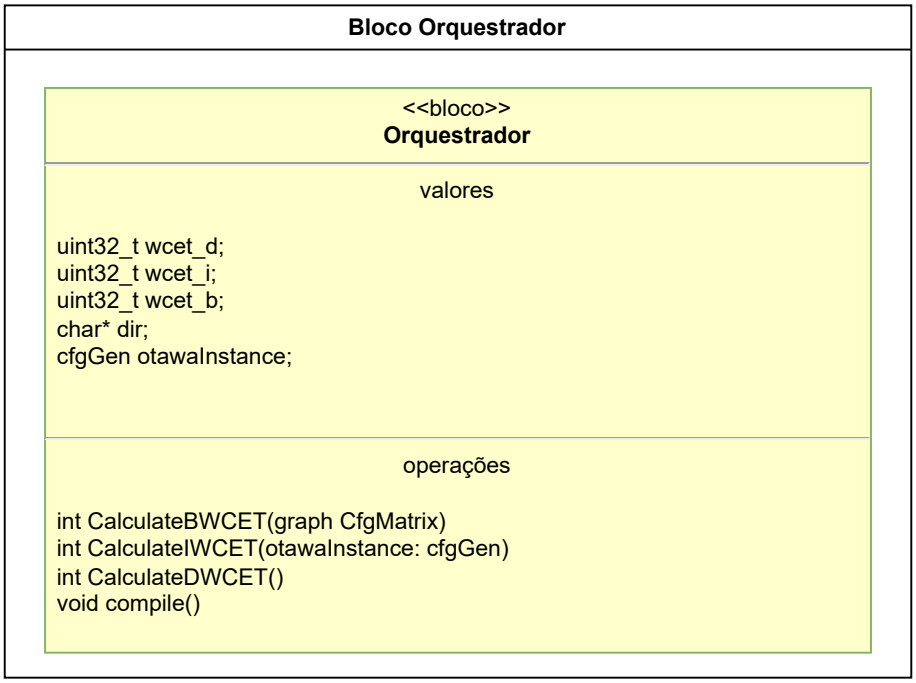

Bloco Dinâmico
<<Bloco>>
Dinâmico
1
wcetDyn
<<bloco>>
WCET_DYN
valores
uint32_t wcet;
std::string scriptPath;
std::string elfPath;
operações
WCETCalculatorDyn(const std::string scriptPath, const std::string elfPath);
void calculateWCET();
uint32_t getWCET();
void flash();
static void startOpenocd();
void endOpenocd();
uint32_t runGDB();
uint32_t findWcetWord(const std::string& texto, const std::string& substring);
1
1
1
1
hwdebug
<<bloco>>
hwdebug
valores
uint32_t n_measures;
list measure;
operações
void findLineCycles();
void mean(int array[])
void main();
1
openocd
st-flash
gdb
gdb
<<Bloco>>
OpenOCD
<<Bloco>>
stlink-tools
<<Bloco>>
GDB


Gerador Grafo
<<Bloco>>
Gerador Grafo
otawaInstance
genff
<<bloco>>
cfgGen
valores
elm::string scriptPath;
elm::string entryFunction;
elm::string elfPath;
otawa::WorkSpace *ws;
operações
cfgGen(const std::string scriptPath, const std::string entryFunction, const std::string elfPath);
std::set<CfgMatrix> cfg2Matrix();
otawa::WorkSpace *workspaceGenerator();
<<bloco>>
generate ff
valores
std::string command;
FILE* in;
operações
void genff();
1
1
mySet
<<bloco>>
CfgMatrix
operações
void setConv(int nodeA, int nodeB, int cycles);
void setLoop(int nodeA, int nodeB, int cycles);
void setIteration(int node, int iterations);
void setFuncCallName(int node, std::string name);
void setMyFunc(std::string name);
void setPriority(int priority);
int getIteration(int node);
std::string getBlockName(int node);
int getCycles(int nodeA, int nodeB);
int getOuts(int node);
int getSumLine(int node);
int getBlockNameHash(int node);
size_t getSize();
std::string getMyName();
int getMyHashName();
int getPriority();
int getMaxCycle();
int isAloop(int node);
void exportCSVs(std::string funcname);
void exportDots(std::string funcname);
void printCycles();
void printFunctions();
void printIterations();
void printOuts();
std::vector<std::vector<int>> get_cycles_from_node(int start_node);
int checkNeastedLoops(std::vector<int> loop);
void printAllLoops(); // imprime todos os ciclos do cfg
int loopHead(int node); // devolve o no que e cabeca de um loop
bool operator<(const CfgMatrix& other)
xml2ff
<<bloco>>
xml2ff
valores
elm::string scriptPath;
elm::string entryFunction;
elm::string elfPath;
otawa::WorkSpace *ws;
operações
void set_ff(const std::string filename, int iterations);
dict extract_xml(const std::string filename);
1
b
otawa
<<Bloco>>
Otawa


Bloco IPET
<<Bloco>>
IPET
wcetIpet
<<bloco>>
WCET_IPET
valores
otawa::WorkSpace *ws;
uint32_t wcet;
operações
WCETCalculator(otawa::WorkSpace *ws);
void calculateWCET();
uint32_t getWCET();
1
ws
<<Bloco>>
Otawa


Bloco ARM GCC
<<Bloco>>
ARM GCC


Bloco Interface
<<bloco>>
Interface
valores
string scriptPath;
string cPath;
string resolvedPath;
bool dyn;
operações
void showUsage(const char* programName)
void setVerbose(bool verbose)
void toFile(std::string dir, std::string entry, uint32_t wceti,
uint32_t wcetb, uint32_t wcetd, bool dyn)


Bloco Orquestrador
<<bloco>>
Orquestrador
valores
uint32_t wcet_d;
uint32_t wcet_i;
uint32_t wcet_b;
char* dir;
cfgGen otawaInstance;
operações
int CalculateBWCET(graph CfgMatrix)
int CalculateIWCET(otawaInstance: cfgGen)
int CalculateDWCET()
void compile()

C. Requisitos do projeto

Requisitos Funcionais de Sistema	
Identificador	**Descrição do requisito**
FRS01	A ferramenta deve receber como entrada um código desenvolvido na linguagem C.
FRS02	A ferramenta deve calcular o WCET utilizando o método IPET, quando a entrada for válida.
FRS03	A ferramenta deve calcular o WCET utilizando método da análise dinâmica, quando a entrada for válida.
FRS04	A ferramenta deve calcular o WCET utilizando método do algoritmo bioinspirado, quando a entrada for válida.
FRS05	A ferramenta deve exibir os resultados de WCET calculados, quando todas as medições forem finalizadas.
FRS06	A ferramenta deve ser capaz de utilizar métodos do OTAWA via API, quando for solicitado.
FRS07	A ferramenta deve ser capaz de ser integrada a uma interface de hardware.
FRS08	A ferramenta deve ser capaz de gerar um arquivo no formato .txt com todos os WCET calculados, quando as medições forem finalizadas.
FRS09	A ferramenta deve utilizar como interface de usuário o prompt de comando para facilitar a entrada de códigos e exibir resultados.
FRS10	A ferramenta deve ser capaz de permitir o usuário inserir o número iterações de loops, quando o módulo de geração de grafo solicitar .
FRS11	A ferramenta deve ter um modo de debug onde mostre por meio de mensagens o passo a passo da execução.
FRS12	A ferramenta deve ter um modo de ajuda que mostre por meio de uma mensagem o funcionamento da ferramenta.
FRS13	A ferramenta deve ser capaz de preencher o número de iterações de loops quando os mesmos forem identificados.
FRS14	A ferramenta deve ser capaz de gerar um grafo de fluxo de controle correspondente ao código de entrada.
FRS15	A ferramenta deve ser capaz de simular o algoritmo ACO.
FRS16	A ferramenta receber como entrada o arquivo que simula a arquitetura do sistema em que o código C será executado.

Requisitos de Alto Nível IPET	
Identificador	**Descrição do requisito**
HRIT01	O módulo IPET deve se conectar à API do OTAWA para calcular o WCET (Worst-Case Execution Time) do código fornecido como entrada usando o método IPET (Implicit Path Enumeration Technique).

Requisitos de Alto Nível Gerador Grafo	
Identificador	**Descrição do requisito**
HRG01	O módulo gerador grafo deve se conectar à API do OTAWA para gerar o grafo de fluxo de controle do código fornecido como entrada.
HRG02	O módulo gerador grafo deve ser capaz de gerar o arquivo .ff a partir do código .elf.
HRG03	O módulo gerador grafo deve ser capaz de gerar o arquivo . elf a partir do código de entrada C.
HRG04	O módulo gerador grafo deve ser capaz de gerar uma matriz que representa a quantidade de iterações de loops a partir do grafo de fluxo de controle.
HRG05	O módulo gerador grafo deve ser capaz de gerar uma matriz que representa as possibilidades de caminhos a partir do grafo de fluxo de controle.

Requisitos de Alto Nível Algoritmo BioInspirado	
Identificador	**Descrição do requisito**
HRB01	O módulo algoritmo deve encontrar o pior caminho do grafo recebido como entrada utilizando o algoritmo ACO.
HRB02	O módulo algoritmo deve ser capaz de atualizar a função fitness de cada formiga.
HRB03	O módulo algoritmo deve considerar todos os possíveis caminhos no grafo durante a busca pelo pior caminho.
HRB04	O módulo algoritmo deve ser capaz de encontrar a melhor função fitness entre todas as formigas.
HRB05	O módulo algoritmo deve ser capaz de fornecer o WCEP, quando receber como entrada um grafo válido.
HRB06	O módulo algoritmo deve ser capaz de atualizar o feromônio baseado no caminho que as formigas realizam.
HRB07	O módulo algoritmo deve ser capaz de simular o algoritmo ACO dada uma quantidade de iterações.

Requisitos de Alto Nível Dinâmico	
Identificador	**Descrição do requisito**
HRD01	O módulo de validação dinâmica deve calcular o WCET a partir do código fornecido como entrada.
HRD02	O módulo de validação dinâmica deve ser capaz de compilar o código de entrada para a arquitetura ARM.
HRD03	O módulo de validação dinâmica deve ser capaz de se conectar ao hardware via OpenOCD.
HRD04	O módulo de validação dinâmica deve ser capaz de executar o código no hardware.
HRD05	O módulo de validação dinâmica deve ser capaz de simular 10 vezes o código de entrada para selecionar o pior tempo de execução.

Requisitos de Alto Nivel Interface	
Identificador	**Descrição do requisito**
HRI01	O módulo de interface deve receber como entrada por meio da chamada da flag "-p" o caminho para um arquivo que contenha o código de entrada.
HRI02	O módulo de interface deve retornar uma mensagem de ajuda caso o caminho para o código de entrada não tenha sido informado.
HRI03	O módulo de interface deve retornar uma mensagem de erro ao usuário caso o código de entrada fornecido não exista.
HRI04	O módulo de interface deve retornar uma mensagem de erro ao usuário caso o arquivo que contenha o código C não tenha extensão .c.
HRI05	O módulo de interface deve ter uma opção de debug por meio da chamada da flag "-v", onde retorna ao usuário, por meio de mensagens, o passo a passo da execução.
HRI06	O módulo de interface deve ter uma opção de ajuda por meio da chamada da flag "-h", que mostra por meio de uma mensagem como a ferramenta pode ser utilizada.
HRI07	O módulo de interface deve ser capaz de gerar um arquivo . txt com os valores WCET gerados pela ferramenta.
HRI08	O módulo de interface deve retornar uma mensagem de erro caso o caminho para o arquivo que descreve a arquitetura utilizando a flag "-s" não seja encontrado.
HRI09	O módulo de interface deve ter uma opção de simular diretamente no hardware por meio da chamada da flag "-d", onde retorna o tempo de execução no hardware.
HRI10	O módulo de interface deve retornar uma mensagem de erro caso a opção de simular diretamente no hardware seja chamada por meio da flag "-d" e o hardware correto não esteja plugado.
HRI11	O módulo de interface deve receber como entrada por meio da chamada da flag "-s" o arquivo que simula a arquitetura do sistema em que o código C será executado.
HRI12	O módulo de interface deve retornar uma mensagem de erro caso o WCET utilizando o método IPET não seja computado.
HRI13	O módulo de interface deve retornar uma mensagem de erro caso o WCET utilizando o método ACO não seja computado.
HRI14	O módulo de interface deve retornar uma mensagem de erro caso o usuário tenha inserido a flag "-d" e o WCET utilizando o método dinâmico não seja computado.

Requisitos Não Funcionais	
Identificador	**Descrição do requisito**
NFR01	A ferramenta deve ter uma documentação sobre seu funcionamento, utilização e configuração.
NFR02	A utilização da API do OTAWA deve estar de acordo com as normas e condições de uso definidor pela API.
NFR03	O algoritmo bio inspirado deve ser selecionado com base na sua eficácia em resolução de problemas que envolvem combinação linar, baixo custo e baixa complexidade computacional.
NFR04	A plataforma de hardware escolhida deve ser compatível com a API do OTAWA.
NFR05	O algoritmo deve ser eficiente em termos de uso de recursos computacionais, como memória e processamento.
NFR06	O módulo de validação dinâmica deve considerar os efeitos de cache do processador no cálculo do WCET.

D. Rastreabilidade

D.1. Rastreabilidade entre requisitos de sistema e alto nível

Requisitos de Sistema	Requisitos Gerador Grafo (HRG)	Requisitos IPET (HRIT)	Requisitos Dinamico (HRD)	Requisitos Bioinspirado (HRB)	Requisitos Interface (HRI)	Requisitos Não Funcionais (NFR)
FRS01	HRG03				HRI01; HRI02; HRI03; HRI04	NFR01
FRS02		HRIT01				NFR01; NFR02
FRS03			HRD01			NFR01; NFR06
FRS04				HRB01; HRB03; HRB04; HRB05		NFR01; NFR03; NFR05
FRS05		HRIT01	HRD01	HRB01		NFR01
FRS06	HRG01	HRIT01			HRI11	NFR01; NFR02; NFR04
FRS07			HRD02; HRD03; HRD04; HRD05		HRI09; HRI10	NFR01; NFR04
FRS08					HRI07	NFR01
FRS09	HRG02; HRG03				HRI01; HRI02; HRI03; HRI04; HRI05; HRI06; HRI08; HRI09; HRI10; HRI12; HRI13; HRI14	NFR01
FRS10	HRG02					NFR01
FRS11					HRI05	NFR01
FRS12					HRI02; HRI06	NFR01
FRS13	HRG04					NFR01
FRS14	HRG04; HRG05					NFR01
FRS15				HRB02; HRB05; HRB06; HRB07		NFR01
FRS16					HRI08; HRI011;	NFR02

D.2. Rastreabilidade entre requisitos de sistema e testes de caixa preta

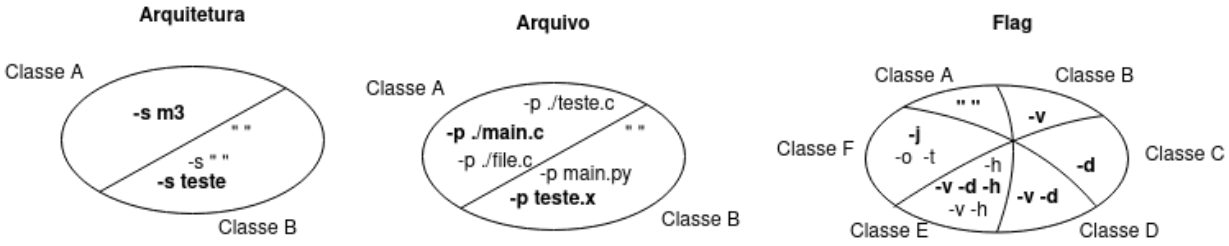
Identificador	Teste de Integração (Caixa Preta)	Quantidade
FRS01	01, 02, 03, 04, 07, 08, 09, 10, 13, 14, 15, 16, 19, 20, 21, 22	16
FRS02	01, 02, 03, 04	4
FRS03	03, 04	2
FRS04	01, 02, 03, 04	4
FRS05	01, 02, 03, 04	4
FRS06	01, 02, 03, 04	4
FRS07	03, 04	2
FRS08	01, 02, 03, 04	4
FRS09	01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24	24
FRS10	01, 02, 03, 04	4
FRS11	02, 04, 08, 10, 14, 16, 20, 22	8
FRS12	05, 06, 11, 12, 17, 18, 23, 24	8
FRS13	01, 02, 03, 04	4
FRS14	01, 02, 03, 04	4
FRS15	01, 02, 03, 04	4
FRS16	01, 02, 03, 04, 07, 08, 09, 10, 13, 14, 15, 16, 19, 20, 21, 22	16

D.3. Rastreabilidade entre requisitos de alto nível, código e testes de caixa branca

Identificador	Funções no Código	Teste Unitário
HRG01	*cfgGen::workspaceGenerator()	WorkSpaceGeneratorTest
HRG02	genff()	GenffTest
HRG03	compile()	CompileTest
HRG04	cfgGen::cfg2Matrix()	CfgToMatrixTest
HRG05	cfgGen::cfg2Matrix()	CfgToMatrixTest
HRIT01	WCETCalculator::getWCET()	WcetIlpetM3Test
HRD01	WCETCalculatorDyn::calculateWCET()	WcetDynamicTest
HRD02	compile()	CompileTest
HRD03	WCETCalculatorDyn::startOpenocd() WCETCalculatorDyn::endOpenocd()	WcetDynamicTest
HRD04	WCETCalculatorDyn::runGDB()	WcetDynamicTest
HRD05	WCETCalculatorDyn::calculateWCET()	WcetDynamicTest
HRB01	WCETCalculatorBio::calculateWCET()	WcetBioM3Test
HRB02	ACO::fitnessFunction(int antNo)	WcetBioM3Test
HRB03	ACO::runColony()	WcetBioM3Test
HRB04	ACO::calculateFitness() ACO::findQueen()	WcetBioM3Test
HRB05	ACO::printWCEP(Ant a)	WcetBioM3Test
HRB06	ACO::updatePhromone()	WcetBioM3Test
HRB07	ACO::simulate(bool verbose)	ACOSimulateTest
HRI01	main(int argc, char* argv[])	CompileTest
HRI02	WCETCalculator::calculateWCET()	WcetIlpetM3Test
HRI03	WCETCalculator::calculateWCET()	WcetBioM3Test
HRI04	WCETCalculator::calculateWCET()	WcetDynamicTest
HRI05	main(int argc, char* argv[]) CfgMatrix::printCycles()	CFGPrintCyclesTest
HRI06	main(int argc, char* argv[]) showUsage(const char* programName)	WcetShowUsageTest
HRI07	toFile()	WriteOutputTest
HRI08	main(int argc, char* argv[]) *cfgGen::workspaceGenerator()	WorkSpaceGeneratorTest
HRI09	main(int argc, char* argv[])	WcetDynamicTest
HRI10	WCETCalculatorDyn::flash()	WcetDynamicTest
HRI11	*cfgGen::workspaceGenerator()	WorkSpaceGeneratorTest
HRI12	main(int argc, char* argv[])	WcetIlpetM3Test
HRI13	main(int argc, char* argv[])	WcetBioM3Test
HRI14	main(int argc, char* argv[])	WcetDynamicTest

E. Casos de teste caixa preta

Classes de equivalência



Variável Arquitetura:
Classe A - Arquitetura existente.
Classe B - Arquitetura não existe.

Variável Arquivo:
Classe A - Arquivo é válido.
Classe B - Arquivo não é válido.

Variável Flag:
Classe A - Flag vazia.
Classe B - Flag verbose (-v).
Classe C - Flag hardware (-d).
Classe D - Combinação flag verbose e flag hardware.
Classe E - Qualquer combinação com a flag ajuda(-h).
Classe F - Flags diferentes de -v, -h e -d.

Mapeamento dos testes

Flag

Classe F	(-s m3, -p ./main.c, -j)	(-s teste, -p ./main.c, -j)	(-s m3, -p teste.x, -j)	(-s teste, -p teste.x, -j)
Classe E	(-s m3, -p ./main.c, -v -d -h)	(-s teste, -p ./main.c, -v -d -h)	(-s m3, -p teste.x, -v -d -h)	(-s teste, -p teste.x, -v -d -h)
Classe D	(-s m3, -p ./main.c, -v -d)	(-s teste, -p ./main.c, -v -d))	(-s m3, -p teste.x, -v -d)	(-s teste, -p teste.x, -v -d)
Classe C	(-s m3, -p ./main.c, -d)	(-s teste, -p ./main.c, -d)	(-s m3, -p teste.x, -d)	(-s teste, -p teste.x, -d)
Classe B	(-s m3, -p ./main.c, -v)	(-s teste, -p ./main.c, -v)	(-s m3, -p teste.x, -v)	(-s teste, -p teste.x, -v)
Classe A	(-s m3, -p ./main.c, __)	(-s teste, -p ./main.c, __)	(-s m3, -p teste.x, __)	(-s teste, -p teste.x, __)
	(A,A)	(B,A)	(A,B)	(B,B)

Arquitetura/Arquivo

Casos de teste caixa preta

SUCESSO	
Caso de Teste	01 - Solicitar cálculo WCET com arquivo e arquitetura válidos, sem flags.
Objetivo	Verificar se o software executa corretamente quando fornecido uma arquitetura válida e um arquivo válido.
Entrada	(A, A, A) Arquitetura: -s m3 Arquivo: -p ./test_files/if/main.c Flag:
Passos	1. Abra o terminal; 2. Digite: *./wcet -s m3 -p ./test_files/if/main.c* ; 3. Aperte enter; 4. Verifique se os valores *WCET_IPET* e *WCET_BIO* são exibidos no terminal; 5. Verifique se o arquivo de saída *results.txt* foi gerado na pasta */test_files/if/* ; 6. Verifique se os valores *WCET_IPET* e *WCET_BIO* de dentro do arquivo */test_files/if/results.txt* são os mesmos que estão exibidos no terminal.
Resultado Esperado	Valores *WCET_IPET* e *WCET_BIO* exibidos em tela e dentro do arquivo *results.txt* .
Requisitos Envolvidos	FRS01, FRS02, FRS04, FRS05, FRS06, FRS08, FRS09, FRS10, FRS13, FRS14, FRS15, FRS16

SUCESSO	
Caso de Teste	02 - Solicitar cálculo WCET com arquivo e arquitetura válidos, com flag verbose.
Objetivo	Verificar se o software executa corretamente quando fornecido uma arquitetura válida, um arquivo válido e a flag verbose.
Entrada	(A, A, B) Arquitetura: -s m3 Arquivo: -p ./test_files/if/main.c Flag: -v
Passos	1. Abra o terminal; 2. Digite: *./wcet -s m3 -p ./test_files/if/main.c -v* ; 3. Aperte enter; 4. Verifique se informações mais detalhadas da execução do software são exibidas no terminal, como por exemplo mudança de diretório, geração de arquivos, compilação, entre outros; 5. Verifique se os valores *WCET_IPET* e *WCET_BIO* são exibidos no terminal; 6. Verifique se o arquivo de saída *results.txt* foi gerado na pasta */test_files/if/* ; 7. Verifique se os valores *WCET_IPET* e *WCET_BIO* de dentro do arquivo */test_files/if/results.txt* são os mesmos que estão exibidos no terminal.
Resultado Esperado	Valores *WCET_IPET*, *WCET_BIO* e *detalhes dos cálculos* exibidos em tela. Os valores estão dentro do arquivo *results.txt* .
Requisitos Envolvidos	FRS01, FRS02, FRS04, FRS05, FRS06, FRS08, FRS09, FRS10, FRS11, FRS13, FRS14, FRS15, FRS16

SUCESSO	
Caso de Teste	03 - Solicitar cálculo WCET com arquivo e arquitetura válidos, com flag hardware.
Objetivo	Verificar se o software executa corretamente quando fornecido uma arquitetura válida, um arquivo válido e a flag hardware.
Entrada	(A, A, C) Arquitetura: -s m3 Arquivo: -p ./test_files/if/main.c Flag: -d
Passos	1. Abra o terminal; 2. Encaixe o hardware *Bluepill Cortex-M3* na entrada USB de sua máquina; 3. Digite: *./wcet -s m3 -p ./test_files/if/main.c -d* ; 4. Aperte enter; 5. Verifique se os valores *WCET_IPET*, *WCET_BIO* e *WCET_DYNAMIC* são exibidos no terminal; 6. Verifique se o arquivo de saída *results.txt* foi gerado na pasta */test_files/if/ ;* 7. Verifique se os valores *WCET_IPET*, *WCET_BIO* e *WCET_DYNAMIC* de dentro do arquivo */test_files/if/results.txt* são os mesmos que estão exibidos no terminal.
Resultado Esperado	Valores *WCET_IPET*, *WCET_BIO* e *WCET_DYNAMIC* exibidos em tela e dentro do arquivo *results.txt* .
Requisitos Envolvidos	FRS01, FRS02, FRS03, FRS04, FRS05, FRS06, FRS07, FRS08, FRS09, FRS10, FRS13, FRS14, FRS15, FRS16

SUCESSO	
Caso de Teste	04 - Solicitar cálculo WCET com arquivo e arquitetura válidos, com flag verbose e hardware.
Objetivo	Verificar se o software executa corretamente quando fornecido uma arquitetura válida, um arquivo válido, a flag verbose e a flag hardware .
Entrada	(A, A, D) Arquitetura: -s m3 Arquivo: -p ./test_files/if/main.c Flag: -v -d
Passos	1. Abra o terminal; 2. Encaixe o hardware *Bluepill Cortex-M3* na entrada USB de sua máquina; 3. Digite: *./wcet -s m3 -p ./test_files/if/main.c -v -d* e aperte ENTER; 4. Verifique se informações mais detalhadas da execução do software são exibidas no terminal, como por exemplo mudança de diretório, geração de arquivos, compilação, entre outros; 5. Verifique se os valores *WCET_IPET*, *WCET_BIO* e *WCET_DYNAMIC* são exibidos no terminal; 6. Verifique se o arquivo de saída *results.txt* foi gerado na pasta */test_files/if/ ;* 7. Verifique se os valores *WCET_IPET*, *WCET_BIO* e *WCET_DYNAMIC* de dentro do arquivo */test_files/if/results.txt* são os mesmos que estão exibidos no terminal.
Resultado Esperado	Valores *WCET_IPET*, *WCET_BIO*, *WCET_DYNAMIC* e detalhes dos cálculos são exibidos em tela. Os valores estão dentro do arquivo *results.txt* .
Requisitos Envolvidos	FRS01, FRS02, FRS03, FRS04, FRS05, FRS06, FRS07, FRS08, FRS09, FRS10, FRS11, FRS13, FRS14, FRS15, FRS16

SUCESSO	
Caso de Teste	05 - Solicitar cálculo WCET com arquivo e arquitetura válidos, com flag verbose, hardware e ajuda.
Objetivo	Verificar se o software exibe o texto de ajuda quando fornecido uma arquitetura válida, um arquivo válido, a flag verbose, a flag hardware e a flag ajuda.
Entrada	(A, A, E) Arquitetura: -s m3 Arquivo: -p ./test_files/if/main.c Flag: -v -d -h
Passos	1. Abra o terminal; 2. Encaixe o hardware *Bluepill Cortex-M3* na entrada USB de sua máquina; 3. Digite: *./wcet -s m3 -p ./test_files/if/main.c -v -d* -h ; 4. Aperte enter; 5. Verifique se informações de como utilizar o software são exibidas no terminal.
Resultado Esperado	Informações de como utilizar o software são exibidas em tela.
Requisitos Envolvidos	FRS09, FRS12

FALHA	
Caso de Teste	06 - Solicitar cálculo WCET com arquivo e arquitetura válidos, com flag inexistente.
Objetivo	Verificar se o software exibe mensagem de opção inválida e texto de ajuda quando fornecido uma arquitetura válida, um arquivo válido e uma flag que não existe.
Entrada	(A, A, F) Arquitetura: -s m3 Arquivo: -p ./test_files/if/main.c Flag: -j
Passos	1. Abra o terminal; 2. Digite: *./wcet -s m3 -p ./test_files/if/main.c -j* ; 3. Aperte enter; 4. Verifique se mostra uma mensagem de *invalid option* no terminal; 5. Verifique se informações de como utilizar o software são exibidas no terminal.
Resultado Esperado	Informações de como utilizar o software são exibidas em tela.
Requisitos Envolvidos	FRS09, FRS12

FALHA	
Caso de Teste	07 - Solicitar cálculo WCET com arquivo válido e arquitetura inválida.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida e um arquivo válido.
Entrada	(B, A, A) Arquitetura: -s teste Arquivo: -p ./test_files/if/main.c Flag:
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/main.c* ; 3. Aperte enter; 4. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS16

FALHA	
Caso de Teste	08 - Solicitar cálculo WCET com arquivo válido e arquitetura inválida, com flag verbose.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida, um arquivo válido e a flag verbose.
Entrada	(B, A, B) Arquitetura: -s teste Arquivo: -p ./test_files/if/main.c Flag: -v
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/main.c -v* ; 3. Aperte enter; 4. Verifique se informações mais detalhadas da execução do software são exibidas no terminal, como por exemplo mudança de diretório, geração de arquivos, compilação, entre outros; 5. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS11, FRS16

FALHA	
Caso de Teste	09 - Solicitar cálculo WCET com arquivo válido e arquitetura inválida, com flag hardware.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida, um arquivo válido e a flag hardware.
Entrada	(B, A, C) Arquitetura: -s teste Arquivo: -p ./test_files/if/main.c Flag: -d
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/main.c -d* ; 3. Aperte enter; 4. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS16

FALHA	
Caso de Teste	10 - Solicitar cálculo WCET com arquivo válido e arquitetura inválida, com flag verbose e hardware.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida, um arquivo válido, a flag verbose e a flag hardware.
Entrada	(B, A, D) Arquitetura: -s teste Arquivo: -p ./test_files/if/main.c Flag: -v -d
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/main.c -v -d* ; 3. Aperte enter; 4. Verifique se informações mais detalhadas da execução do software são exibidas no terminal, como por exemplo mudança de diretório, geração de arquivos, compilação, entre outros; 5. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS11, FRS16

SUCESSO	
Caso de Teste	11 - Solicitar cálculo WCET com arquivo válido e arquitetura inválida, com flag verbose, hardware e ajuda.
Objetivo	Verificar se o software exibe mensagem de ajuda quando fornecido uma arquitetura inválida, um arquivo válido, a flag verbose, flag hardware e flag ajuda.
Entrada	(B, A, E) Arquitetura: -s teste Arquivo: -p ./test_files/if/main.c Flag: -v -d -h
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/main.c -v -d -h* ; 3. Aperte enter; 4. Verifique se informações de como utilizar o software são exibidas no terminal.
Resultado Esperado	Informações de como utilizar o software são exibidas em tela.
Requisitos Envolvidos	FRS09, FRS12

FALHA	
Caso de Teste	12 - Solicitar cálculo WCET com arquivo válido e arquitetura inválida, com flag inválida.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida, um arquivo válido e uma flag inválida.
Entrada	(B, A, F) Arquitetura: -s teste Arquivo: -p ./test_files/if/main.c Flag: -j
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/main.c -j* ; 3. Aperte enter; 4. Verifique se mostra uma mensagem de *invalid option* no terminal; 5. Verifique se informações de como utilizar o software são exibidas no terminal.
Resultado Esperado	Informações de como utilizar o software são exibidas em tela.
Requisitos Envolvidos	FRS09, FRS12

FALHA	
Caso de Teste	13 - Solicitar cálculo WCET com arquivo inválido e arquitetura válida.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura válida e um arquivo inválido.
Entrada	(A, B, A) Arquitetura: -s m3 Arquivo: -p ./test_files/if/teste.x Flag:
Passos	1. Abra o terminal; 2. Digite: *./wcet -s* m3 *-p ./test_files/if/*teste.x ; 3. Aperte enter; 4. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS16

FALHA	
Caso de Teste	14 - Solicitar cálculo WCET com arquivo inválido e arquitetura válida, com flag verbose.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura válida, um arquivo inválido e a flag verbose.
Entrada	(A, B, B) Arquitetura: -s m3 Arquivo: -p ./test_files/if/teste.x Flag: -v
Passos	1. Abra o terminal; 2. Digite: *./wcet -s* m3 *-p ./test_files/if/*teste.x *-v* ; 3. Aperte enter; 4. Verifique se informações mais detalhadas da execução do software são exibidas no terminal, como por exemplo mudança de diretório, geração de arquivos, compilação, entre outros; 5. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS11, FRS16

FALHA	
Caso de Teste	15 - Solicitar cálculo WCET com arquivo inválido e arquitetura válida, com flag hardware.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura válida, um arquivo inválido e a flag hardware.
Entrada	(A,B,C) Arquitetura: -s m3 Arquivo: -p ./test_files/if/teste.x Flag: -d
Passos	1. Abra o terminal; 2. Digite: *./wcet -s* m3 *-p ./test_files/if/*teste.x *-d* ; 3. Aperte enter; 4. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS16

FALHA	
Caso de Teste	16 - Solicitar cálculo WCET com arquivo inválido e arquitetura válida, com flag verbose e hardware..
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura válida, um arquivo inválido, flag verbose e a flag hardware.
Entrada	(A, B, D) Arquitetura: -s m3 Arquivo: -p ./test_files/if/teste.x Flag: -v -d
Passos	1. Abra o terminal; 2. Digite: *./wcet -s* m3 *-p ./test_files/if/*teste.x *-v -d* ; 3. Aperte enter; 4. Verifique se informações mais detalhadas da execução do software são exibidas no terminal, como por exemplo mudança de diretório, geração de arquivos, compilação, entre outros; 5. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS11, FRS16

SUCESSO	
Caso de Teste	17 - Solicitar cálculo WCET com arquivo inválido e arquitetura válida, com flag verbose, hardware e ajuda.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura válida, um arquivo inválido, flag verbose, flag hardware e a flag ajuda.
Entrada	(A, B, E) Arquitetura: -s m3 Arquivo: -p ./test_files/if/teste.x Flag: -v -d -h
Passos	1. Abra o terminal; 2. Digite: *./wcet -s* m3 *-p ./test_files/if/*teste.x *-v -d -h* ; 3. Aperte enter; 4. Verifique se informações de como utilizar o software são exibidas no terminal.
Resultado Esperado	Informações de como utilizar o software são exibidas em tela.
Requisitos Envolvidos	FRS09, FRS12

FALHA	
Caso de Teste	18 - Solicitar cálculo WCET com arquivo inválido e arquitetura válida, com flag inválida.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura válida, um arquivo inválido e uma flag inválida.
Entrada	(A, B, F) Arquitetura: -s m3 Arquivo: -p ./test_files/if/teste.x Flag: -j
Passos	1. Abra o terminal; 2. Digite: *./wcet -s* m3 *-p ./test_files/if/*teste.x *-j* ; 3. Aperte enter; 4. Verifique se mostra uma mensagem de *invalid option* no terminal; 5. Verifique se informações de como utilizar o software são exibidas no terminal.
Resultado Esperado	Informações de como utilizar o software são exibidas em tela.
Requisitos Envolvidos	FRS09, FRS12

FALHA	
Caso de Teste	19 - Solicitar cálculo WCET com arquivo e arquitetura inválidos.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida e um arquivo inválido.
Entrada	(B, B, A) Arquitetura: -s teste Arquivo: -p ./test_files/if/teste.x Flag:
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/*teste.x ; 3. Aperte enter; 4. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS16

FALHA	
Caso de Teste	20 - Solicitar cálculo WCET com arquivo e arquitetura inválidos, com flag verbose.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida, um arquivo inválido e a flag verbose.
Entrada	(B, B, B) Arquitetura: -s teste Arquivo: -p ./test_files/if/teste.x Flag: -v
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/*teste.x *-v* ; 3. Aperte enter; 4. Verifique se informações mais detalhadas da execução do software são exibidas no terminal, como por exemplo mudança de diretório, geração de arquivos, compilação, entre outros; 5. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS11, FRS16

FALHA	
Caso de Teste	21 - Solicitar cálculo WCET com arquivo e arquitetura inválidos, com flag hardware.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida, um arquivo inválido e a flag hardware.
Entrada	(B, B, C) Arquitetura: -s teste Arquivo: -p ./test_files/if/teste.x Flag: -d
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/*teste.x *-d* ; 3. Aperte enter; 4. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS16

FALHA	
Caso de Teste	22 - Solicitar cálculo WCET com arquivo e arquitetura inválidos, com flag verbose e hardware..
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida, um arquivo inválido, a flag verbose e a flag hardware.
Entrada	(B, B, D) Arquitetura: -s teste Arquivo: -p ./test_files/if/teste.x Flag: -v -d
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/*teste.x *-v -d* ; 3. Aperte enter; 4. Verifique se informações mais detalhadas da execução do software são exibidas no terminal, como por exemplo mudança de diretório, geração de arquivos, compilação, entre outros; 5. Verifique se mostra uma mensagem de erro no terminal.
Resultado Esperado	O software exibe um erro na tela.
Requisitos Envolvidos	FRS01, FRS09, FRS11, FRS16

SUCESSO	
Caso de Teste	23 - Solicitar cálculo WCET com arquivo e arquitetura inválidos, com flag verbose, hardware e ajuda.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida, um arquivo inválido, a flag verbose,a flag hardware e a flag ajuda.
Entrada	(B, B, E) Arquitetura: -s teste Arquivo: -p ./test_files/if/teste.x Flag: -v -d -h
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/*teste.x *-v -d -h* ; 3. Aperte enter; 4. Verifique se informações de como utilizar o software são exibidas no terminal.
Resultado Esperado	Informações de como utilizar o software são exibidas em tela.
Requisitos Envolvidos	FRS09, FRS12

FALHA	
Caso de Teste	24 - Solicitar cálculo WCET com arquivo e arquitetura inválidos, com flag inválida.
Objetivo	Verificar se o software exibe mensagem de erro quando fornecido uma arquitetura inválida, um arquivo inválido, a flag verbose e a flag hardware.
Entrada	(B, B ,F) Arquitetura: -s teste Arquivo: -p ./test_files/if/teste.x Flag: -j
Passos	1. Abra o terminal; 2. Digite: *./wcet -s teste -p ./test_files/if/*teste.x *-j ;* 3. Aperte enter; 4. Verifique se mostra uma mensagem de *invalid option* no terminal; 5. Verifique se informações de como utilizar o software são exibidas no terminal.
Resultado Esperado	Informações de como utilizar o software são exibidas em tela.
Requisitos Envolvidos	FRS09, FRS12

F. Casos de teste caixa branca

Casos de teste caixa branca

Caso de Teste	01 - WorkSpaceGeneratorTest
Objetivo	Verificar a funcionalidade de gerar um workspace do OTAWA
Entrada	Arquitetura utilizada; Caminho do algoritmo a ser analisado; Função de entrada
Resultado esperado	É esperado o retorno de um workspace do OTAWA.
Requisitos Envolvidos	HRG01, HRI08, HRI11

Caso de Teste	02 - CompileTest
Objetivo	Verificar a funcionalidade de compilar o arquivo de entrada para os padrões legíveis pelo OTAWA através do comando make
Entrada	Caminho do arquivo que será feito o teste; Comando make;
Resultado esperado	É esperada a criação dos arquivos base para análise do OTAWA em /din/build
Requisitos Envolvidos	HRG03, HRD02, HRI01

Caso de Teste	03 - GenffTest
Objetivo	Verificar a funcionalidade de gerar o arquivo main_otawa.ff
Entrada	xml gerado pela função CompileTest
Resultado esperado	É esperada a criação do arquivo main_otawa.ff na pasta /din/build
Requisitos Envolvidos	HRG02

Caso de Teste	04 - CfgToMatrixTest
Objetivo	Verificar a funcionalidade de gerar uma matriz a partir do CFG do algoritmo
Entrada	Workspace válido do OTAWA
Resultado esperado	É esperado como retorno um o objeto do tipo cfgMatrix
Requisitos Envolvidos	HRG04, HRG05

Caso de Teste	05 - WcetIipetM3Test
Objetivo	Verificar a funcionalidade do sistema de calcular o WCET pelo método do IPET
Entrada	Workspace válido do OTAWA

Resultado esperado	É esperado o valor do WCET calculado pelo método do IPET
Requisitos Envolvidos	HRIT01, HRI02, HRI12

Caso de Teste	06 - WcetDynamicTest
Objetivo	Verificar a funcionalidade do sistema de calcular o WCET pelo método do dinâmico
Entrada	Arquivo .elf do algoritmo a ser testado
Resultado esperado	É esperado o valor do WCET calculado pelo método dinâmico
Requisitos Envolvidos	HRD01, HRD03, HRD04, HRD05, HRI04, HRI09, HRI10, HRI14

Caso de Teste	07 - WcetBioM3Test
Objetivo	Verificar a funcionalidade do sistema de calcular o WCET pelo método bioinspirado
Entrada	Objeto do tipo CfgMatrix
Resultado esperado	É esperado o valor do WCET calculado pelo método do bioinspirado
Requisitos Envolvidos	HRB01, HRB02, HRB03, HRB04, HRB05, HRB06, HRI03, HRI13

Caso de Teste	08 - ACOSimulateTest
Objetivo	Verificar a funcionalidade do sistema de simular o algoritmo ACO dada uma quantidade de iterações
Entrada	Objeto do tipo cfgMatrix; Quantidade de formiga; Primeiro nó; Máximo de iterações; Alpha; Beta; rho
Resultado esperado	É esperado o valor do WCET calculado pelo método do bioinspirado
Requisitos Envolvidos	HRB01, HRB07

Caso de Teste	09 - CFGPrintCyclesTest
Objetivo	Verificar a funcionalidade do sistema imprimir os ciclos das iterações
Entrada	Objeto do tipo cfgMatrix; Quantidade de formiga; Primeiro nó; Máximo de iterações; Alpha; Beta;

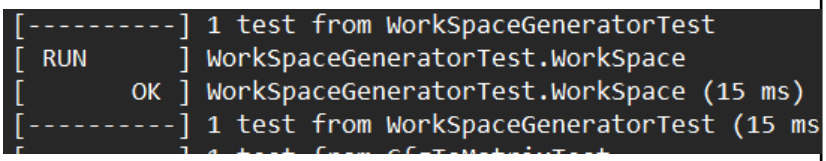
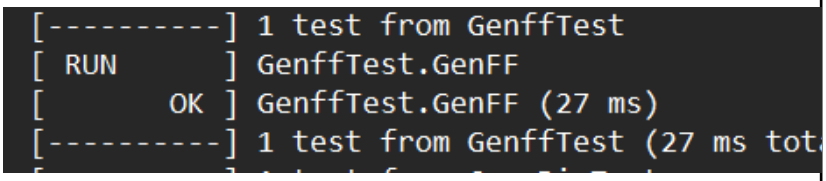
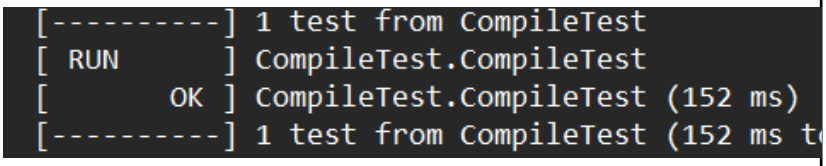
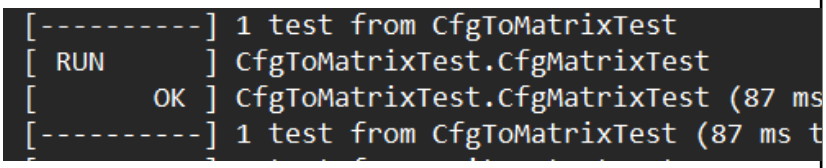
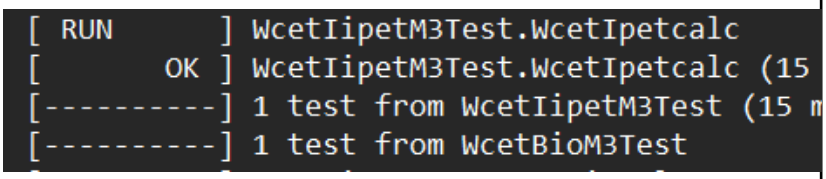
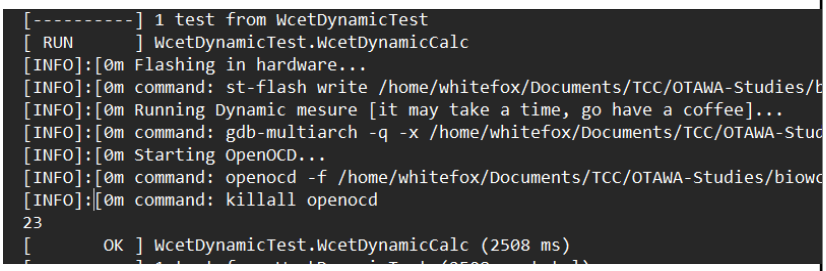
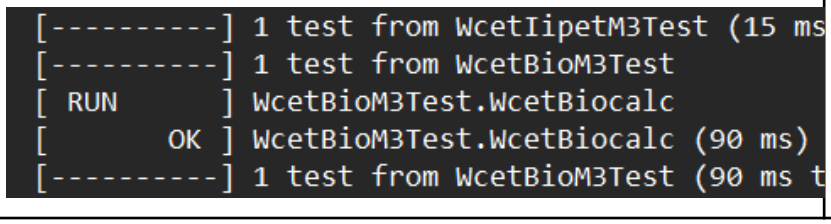
	rho
Resultado esperado	É esperado a visualização da matriz de iterações efetuada pelo software
Requisitos Envolvidos	HRI05

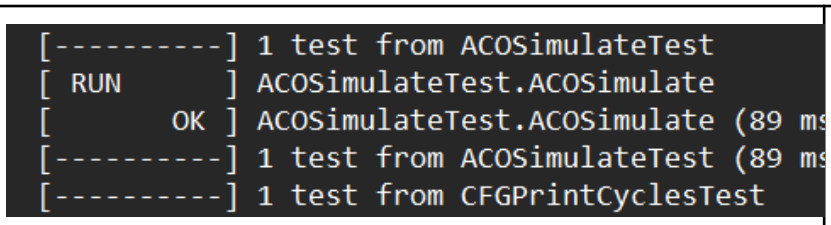
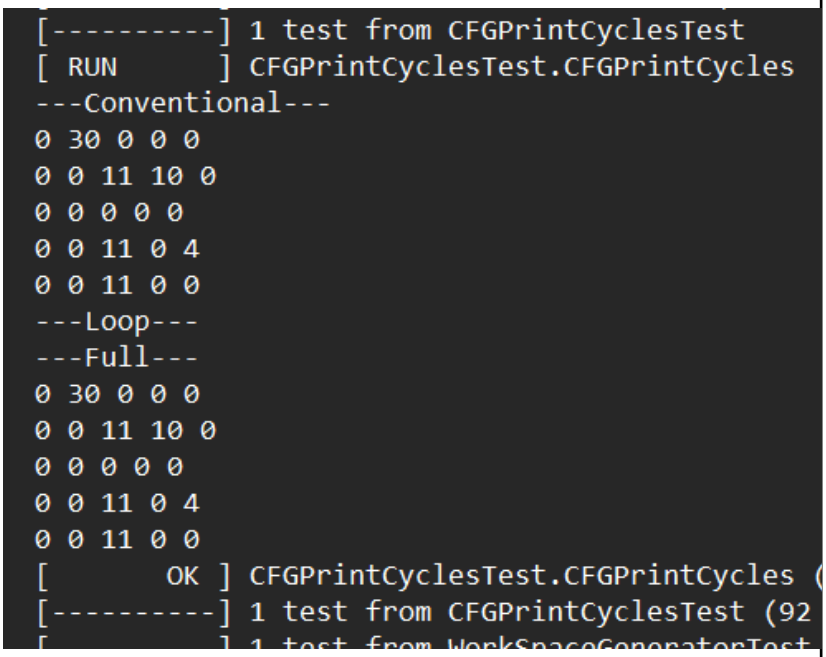
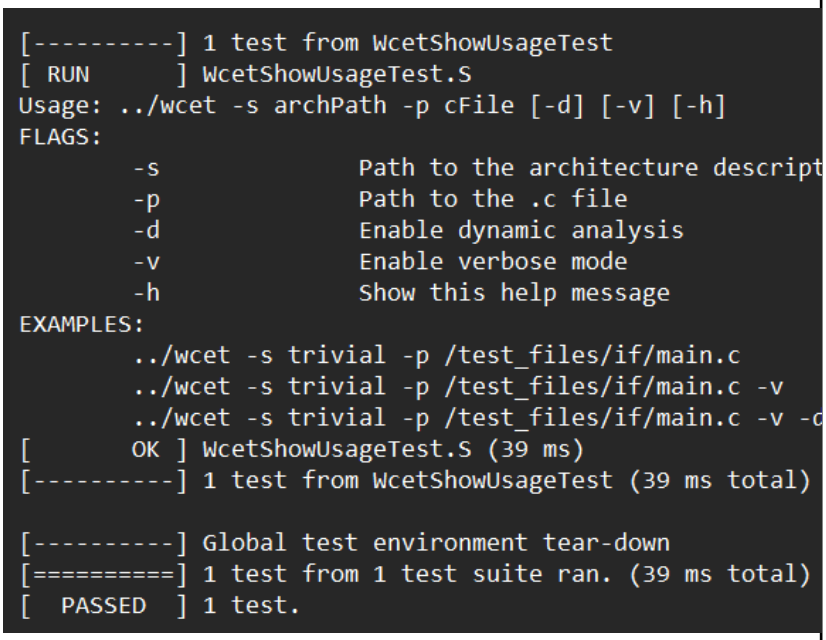
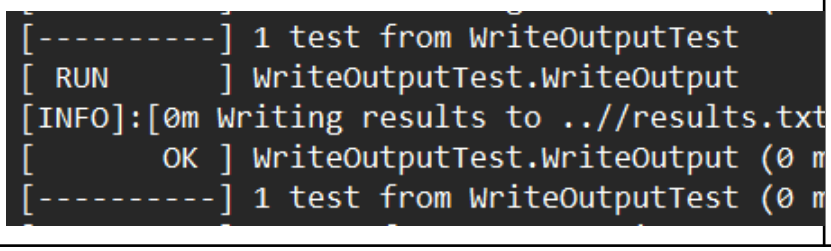
Caso de Teste	10 - WcetShowUsageTest
Objetivo	Verificar a funcionalidade do sistema imprimir o menu com a ajuda de uso
Entrada	Chamada do software sem nenhum parâmetro passado
Resultado esperado	É esperada a visualização do menu de ajuda do software
Requisitos Envolvidos	HRI06

Caso de Teste	11 - WriteOutputTest
Objetivo	Verificar a funcionalidade do sistema de salvar em arquivo os valores dos teste de WCET
Entrada	Valores de WCET calculados
Resultado esperado	É esperada a criação de um arquivo na pasta do algoritmo analisado com os resultados do teste
Requisitos Envolvidos	HRI07

G. Resultados dos testes de caixa branca

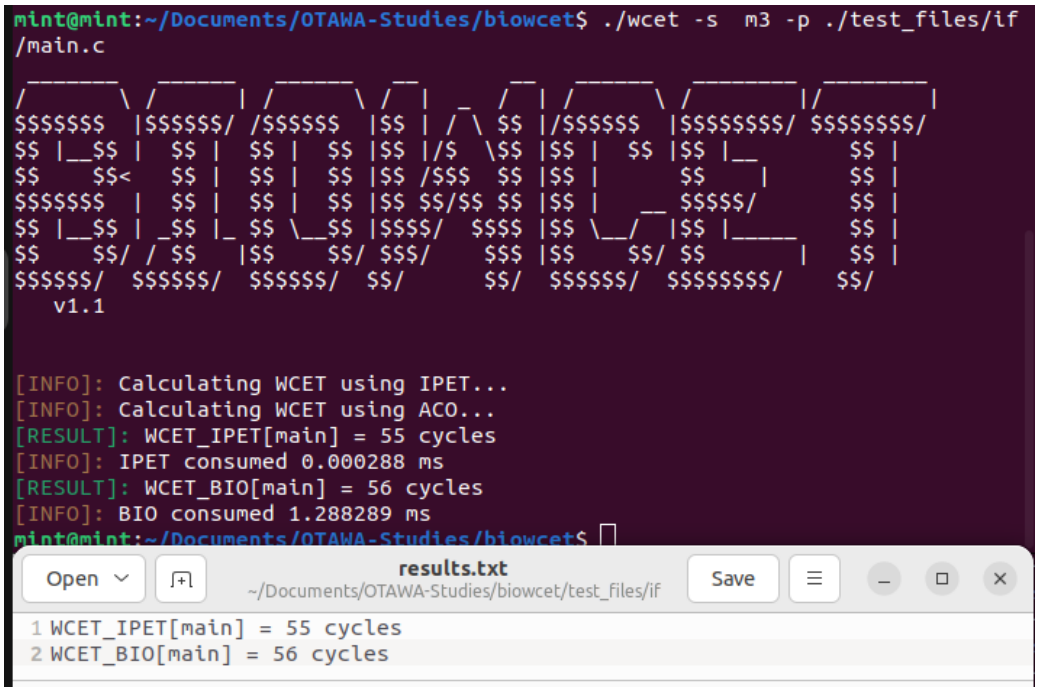
Resultados dos testes de caixa branca

Caso de teste	Resultado	Status
WorkSpaceGeneratorTest		OK
GenffTest		OK
CompileTest		OK
CfgToMatrixTest		OK
WcetIipetM3Test		OK
WcetDynamicTest		OK
WcetBioM3Test		OK

ACOSimulateTest		**OK**
CFGPrintCyclesTest		**OK**
WcetShowUsageTest		**OK**
WriteOutputTest		**OK**

H. Resultados dos testes de caixa preta

Resultados dos testes de caixa preta

Caso de teste	Resultado
01	

02

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s  m3 -p ./test_files/if/main.c -v
 _______  ______  _______  __       __  ______  ________ ________
/       \ /      | /       \ /  |  _  /  | /      \ /        |/        |
$$$$$$$  |$$$$$$/ /$$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[INFO]: Cleaning directory
[INFO]: Directory changed
[INFO]: command: make clean
[INFO]: command: cp './test_files/if/main.c' '/home/mint/Documents/OTAWA-Studies/biowcet/din/s
rc/main.c'
[INFO]: command: realpath ./test_files/if/main.c
[INFO]: Compiling...
[INFO]: Directory changed
[INFO]: command: make
[INFO]: Smart Generating of ff file...
[INFO]: Calculating WCET using IPET...
[INFO]: Calculating WCET using ACO...
---Conventional---
0 31 0 0 0
0 0 11 10 0
0 0 0 0 0
0 0 11 0 4
0 0 11 0 0
---Loop---
---Full---
0 31 0 0 0
0 0 11 10 0
0 0 0 0 0
0 0 11 0 4
0 0 11 0 0
[RESULT]: WCET_IPET[main] = 55 cycles
[INFO]: IPET consumed 0.000735 ms
[RESULT]: WCET_BIO[main] = 56 cycles
[INFO]: BIO consumed 1.249432 ms
[INFO]: Writing results to /home/mint/Documents/OTAWA-Studies/biowcet/test_files/if/results.tx
t
mint@mint:~/Documents/OTAWA-Studies/biowcet$
```

Open ~/Documents/OTAWA-Studies/biowcet/test_files/if **results.txt** Save

```
1 WCET_IPET[main] = 55 cycles
2 WCET_BIO[main] = 56 cycles
```

Plain Text Tab Width: 8 Ln 2, Col 27 INS

03

```
gugagop@gugagop-Lenovo-IdeaPad-S145-15API:~/Documents/PES/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/main.c -d
 _______  ______  _______  __       __  ______  ________ ________
/       \ /      | /       \ /  |  _  /  | /      \ /        |/        |
$$$$$$$  |$$$$$$/ /$$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[INFO]: Calculating WCET using IPET...
[INFO]: Calculating WCET using ACO...
[INFO]: Calculating Execution Time using the Hardware...
[RESULT]: WCET_IPET[main] = 55 cycles
[INFO]: IPET consumed 0.001885 ms
[RESULT]: WCET_BIO[main] = 56 cycles
[INFO]: BIO consumed 12.909506 ms
[RESULT]: WCET_DYNAMIC[main] = 28 cycles
[INFO]: DYNAMIC consumed 2287.792996 ms
```

results.txt (~/Documents/PES/OTAWA-Studies/biowcet/test_files/if)

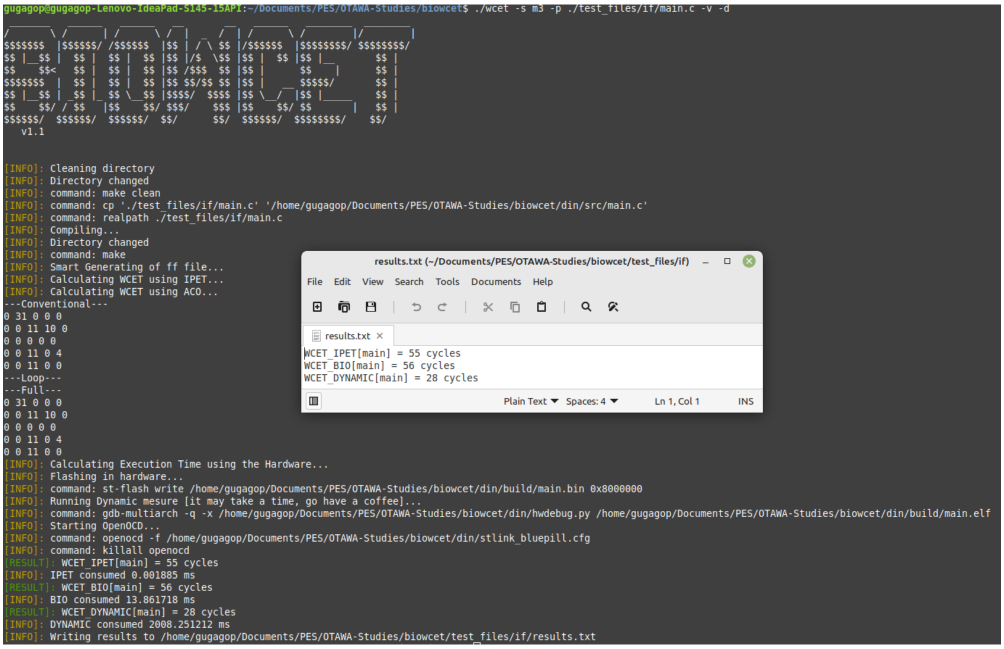
File Edit View Search Tools Documents Help

results.txt

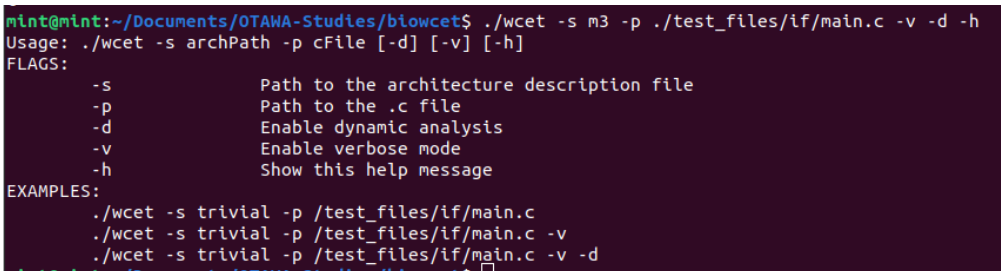
```
WCET_IPET[main] = 55 cycles
WCET_BIO[main] = 56 cycles
WCET_DYNAMIC[main] = 28 cycles
```

Plain Text Spaces: 4 Ln 1, Col 1 INS

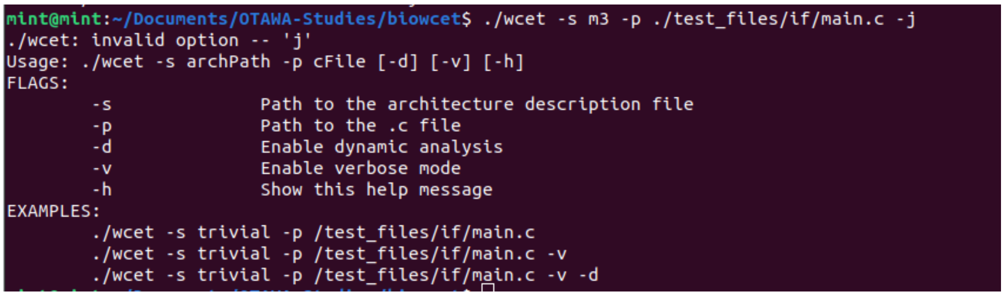
04


gugagop@gugagop-Lenovo-IdeaPad-S145-15API:~/Documents/PES/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/main.c -v -d
v1.1
[INFO]: Cleaning directory
[INFO]: Directory changed
[INFO]: command: make clean
[INFO]: command: cp './test_files/if/main.c' '/home/gugagop/Documents/PES/OTAWA-Studies/biowcet/din/src/main.c'
[INFO]: command: realpath ./test_files/if/main.c
[INFO]: Compiling...
[INFO]: Directory changed
[INFO]: command: make
[INFO]: Smart Generating of ff file...
[INFO]: Calculating WCET using IPET...
[INFO]: Calculating WCET using ACO...
---Conventional---
0 31 0 0 0
0 0 11 10 0
0 0 0 0 0
0 0 11 0 4
0 0 11 0 0
---Loop---
---Full---
0 31 0 0 0
0 0 11 10 0
0 0 0 0 0
0 0 11 0 4
0 0 11 0 0
[INFO]: Calculating Execution Time using the Hardware...
[INFO]: Flashing in hardware...
[INFO]: command: st-flash write /home/gugagop/Documents/PES/OTAWA-Studies/biowcet/din/build/main.bin 0x8000000
[INFO]: Running Dynamic mesure [it may take a time, go have a coffee]...
[INFO]: command: gdb-multiarch -q -x /home/gugagop/Documents/PES/OTAWA-Studies/biowcet/din/hwdebug.py /home/gugagop/Documents/PES/OTAWA-Studies/biowcet/din/build/main.elf
[INFO]: Starting OpenOCD...
[INFO]: command: openocd -f /home/gugagop/Documents/PES/OTAWA-Studies/biowcet/din/stlink_bluepill.cfg
[INFO]: command: killall openocd
[RESULT]: WCET_IPET[main] = 55 cycles
[INFO]: IPET consumed 0.001885 ms
[RESULT]: WCET_BIO[main] = 56 cycles
[INFO]: BIO consumed 13.861718 ms
[RESULT]: WCET_DYNAMIC[main] = 28 cycles
[INFO]: DYNAMIC consumed 2008.251212 ms
[INFO]: Writing results to /home/gugagop/Documents/PES/OTAWA-Studies/biowcet/test_files/if/results.txt
results.txt (~/Documents/PES/OTAWA-Studies/biowcet/test_files/if)
File Edit View Search Tools Documents Help
results.txt
WCET_IPET[main] = 55 cycles
WCET_BIO[main] = 56 cycles
WCET_DYNAMIC[main] = 28 cycles
Plain Text Spaces: 4 Ln 1, Col 1 INS

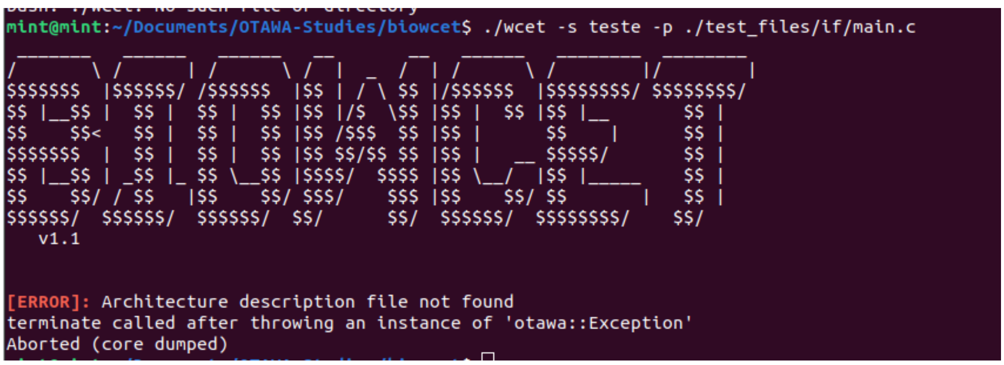
05


mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/main.c -v -d -h
Usage: ./wcet -s archPath -p cFile [-d] [-v] [-h]
FLAGS:
-s Path to the architecture description file
-p Path to the .c file
-d Enable dynamic analysis
-v Enable verbose mode
-h Show this help message
EXAMPLES:
./wcet -s trivial -p /test_files/if/main.c
./wcet -s trivial -p /test_files/if/main.c -v
./wcet -s trivial -p /test_files/if/main.c -v -d

06


mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/main.c -j
./wcet: invalid option -- 'j'
Usage: ./wcet -s archPath -p cFile [-d] [-v] [-h]
FLAGS:
-s Path to the architecture description file
-p Path to the .c file
-d Enable dynamic analysis
-v Enable verbose mode
-h Show this help message
EXAMPLES:
./wcet -s trivial -p /test_files/if/main.c
./wcet -s trivial -p /test_files/if/main.c -v
./wcet -s trivial -p /test_files/if/main.c -v -d

07


mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/main.c
v1.1
[ERROR]: Architecture description file not found
terminate called after throwing an instance of 'otawa::Exception'
Aborted (core dumped)

08

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/main.c -v
 _______  ______  ______  __      __  ______  _______ _______
/       \/      |/      \/  |  _  /  |/      \/       /       |
$$$$$$$  |$$$$$$/ /$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[INFO]: Cleaning directory
[INFO]: Directory changed
[INFO]: command: make clean
[INFO]: command: cp './test_files/if/main.c' '/home/mint/Documents/OTAWA-Studies/biowcet/din/s
rc/main.c'
[INFO]: command: realpath ./test_files/if/main.c
[INFO]: Compiling...
[INFO]: Directory changed
[INFO]: command: make
[INFO]: Smart Generating of ff file...
[ERROR]: Architecture description file not found
terminate called after throwing an instance of 'otawa::Exception'
Aborted (core dumped)
```

09

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/main.c -d
 _______  ______  ______  __      __  ______  _______ _______
/       \/      |/      \/  |  _  /  |/      \/       /       |
$$$$$$$  |$$$$$$/ /$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[ERROR]: Architecture description file not found
terminate called after throwing an instance of 'otawa::Exception'
Aborted (core dumped)
```

10

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/main.c -v -d
 _______  ______  ______  __      __  ______  _______ _______
/       \/      |/      \/  |  _  /  |/      \/       /       |
$$$$$$$  |$$$$$$/ /$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[INFO]: Cleaning directory
[INFO]: Directory changed
[INFO]: command: make clean
[INFO]: command: cp './test_files/if/main.c' '/home/mint/Documents/OTAWA-Studies/biowcet/din/s
rc/main.c'
[INFO]: command: realpath ./test_files/if/main.c
[INFO]: Compiling...
[INFO]: Directory changed
[INFO]: command: make
[INFO]: Smart Generating of ff file...
[ERROR]: Architecture description file not found
terminate called after throwing an instance of 'otawa::Exception'
Aborted (core dumped)
```

11

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/main.c -v -d -
h
Usage: ./wcet -s archPath -p cFile [-d] [-v] [-h]
FLAGS:
        -s              Path to the architecture description file
        -p              Path to the .c file
        -d              Enable dynamic analysis
        -v              Enable verbose mode
        -h              Show this help message
EXAMPLES:
        ./wcet -s trivial -p /test_files/if/main.c
        ./wcet -s trivial -p /test_files/if/main.c -v
        ./wcet -s trivial -p /test_files/if/main.c -v -d
```

12

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/main.c -j
./wcet: invalid option -- 'j'
Usage: ./wcet -s archPath -p cFile [-d] [-v] [-h]
FLAGS:
        -s              Path to the architecture description file
        -p              Path to the .c file
        -d              Enable dynamic analysis
        -v              Enable verbose mode
        -h              Show this help message
EXAMPLES:
        ./wcet -s trivial -p /test_files/if/main.c
        ./wcet -s trivial -p /test_files/if/main.c -v
        ./wcet -s trivial -p /test_files/if/main.c -v -d
```

13

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/teste.x
 _______  ______  ______  __      __  ______  _______ _______
/       \ /      | /      \ /  |  _  /  | /      \ /       |/       |
$$$$$$$  |$$$$$$/ /$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[ERROR]: Error in: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main
.c' 2>&1
[ERROR]: cp: cannot stat './test_files/if/teste.x': No such file or directory
```

14

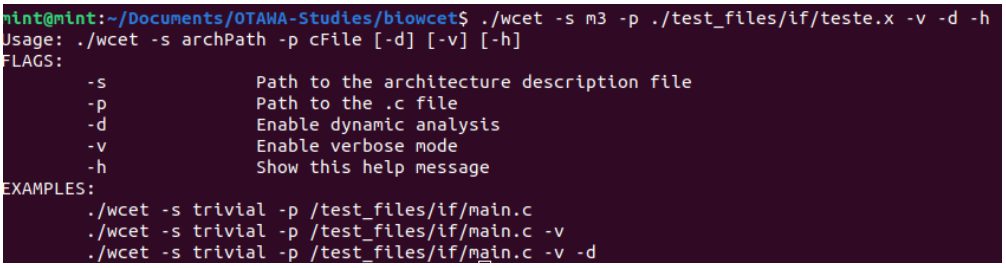
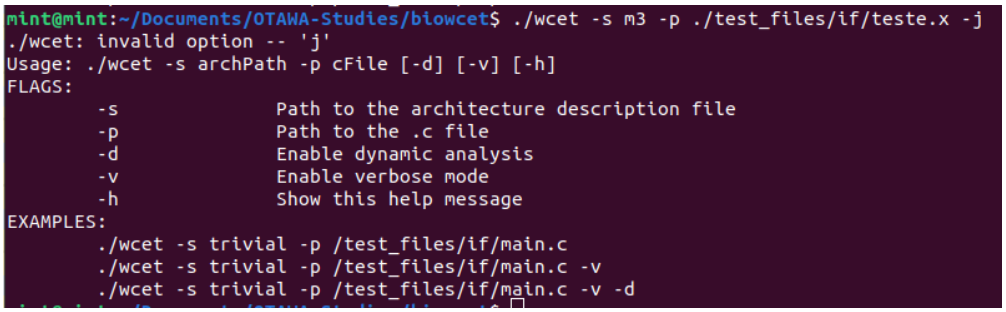
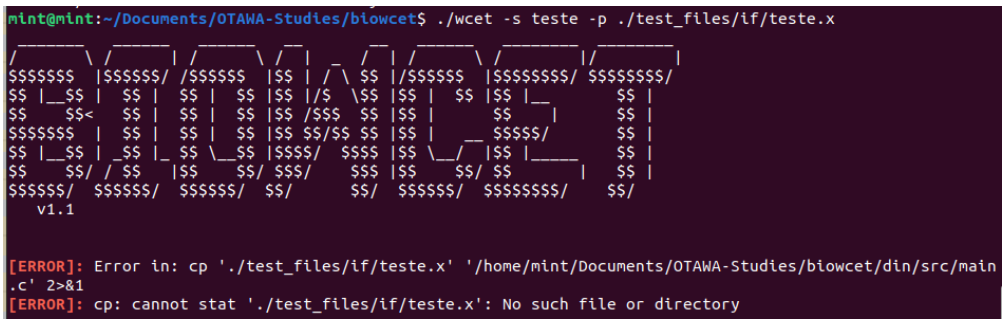
```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/teste.x -v
 _______  ______  ______  __      __  ______  _______ _______
/       \ /      | /      \ /  |  _  /  | /      \ /       |/       |
$$$$$$$  |$$$$$$/ /$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[INFO]: Cleaning directory
[INFO]: Directory changed
[INFO]: command: make clean
[INFO]: command: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main.c
'
[ERROR]: Error in: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main
.c' 2>&1
[ERROR]: cp: cannot stat './test_files/if/teste.x': No such file or directory
```

15

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/teste.x -d
 _______  ______  ______  __      __  ______  _______ _______
/       \ /      | /      \ /  |  _  /  | /      \ /       |/       |
$$$$$$$  |$$$$$$/ /$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[ERROR]: Error in: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main
.c' 2>&1
[ERROR]: cp: cannot stat './test_files/if/teste.x': No such file or directory
```

16	 mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/teste.x -v -d v1.1 [INFO]: Cleaning directory [INFO]: Directory changed [INFO]: command: make clean [INFO]: command: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main.c ' [ERROR]: Error in: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main .c' 2>&1 [ERROR]: cp: cannot stat './test_files/if/teste.x': No such file or directory
17	 mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/teste.x -v -d -h Usage: ./wcet -s archPath -p cFile [-d] [-v] [-h] FLAGS: -s Path to the architecture description file -p Path to the .c file -d Enable dynamic analysis -v Enable verbose mode -h Show this help message EXAMPLES: ./wcet -s trivial -p /test_files/if/main.c ./wcet -s trivial -p /test_files/if/main.c -v ./wcet -s trivial -p /test_files/if/main.c -v -d
18	 mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s m3 -p ./test_files/if/teste.x -j ./wcet: invalid option -- 'j' Usage: ./wcet -s archPath -p cFile [-d] [-v] [-h] FLAGS: -s Path to the architecture description file -p Path to the .c file -d Enable dynamic analysis -v Enable verbose mode -h Show this help message EXAMPLES: ./wcet -s trivial -p /test_files/if/main.c ./wcet -s trivial -p /test_files/if/main.c -v ./wcet -s trivial -p /test_files/if/main.c -v -d
19	 mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/teste.x v1.1 [ERROR]: Error in: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main .c' 2>&1 [ERROR]: cp: cannot stat './test_files/if/teste.x': No such file or directory

20

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/teste.x -v
 _______   ______   ______   __       __   ______   ________ ________
/       \ /      | /      \ /  |  _   /  | /      \ /        |/        |
$$$$$$$  |$$$$$$/ /$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[INFO]: Cleaning directory
[INFO]: Directory changed
[INFO]: command: make clean
[INFO]: command: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main.c
'
[ERROR]: Error in: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main
.c' 2>&1
[ERROR]: cp: cannot stat './test_files/if/teste.x': No such file or directory
```

21

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/teste.x -d
 _______   ______   ______   __       __   ______   ________ ________
/       \ /      | /      \ /  |  _   /  | /      \ /        |/        |
$$$$$$$  |$$$$$$/ /$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[ERROR]: Error in: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main
.c' 2>&1
[ERROR]: cp: cannot stat './test_files/if/teste.x': No such file or directory
```

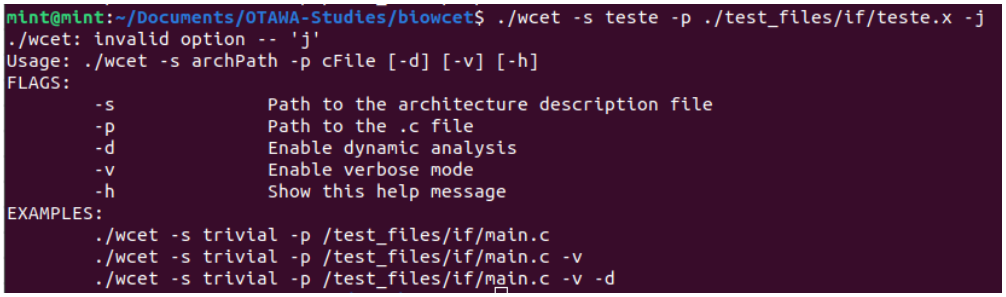
22

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/teste.x -v -d
 _______   ______   ______   __       __   ______   ________ ________
/       \ /      | /      \ /  |  _   /  | /      \ /        |/        |
$$$$$$$  |$$$$$$/ /$$$$$$  |$$ | / \ $$ |/$$$$$$  |$$$$$$$$/ $$$$$$$$/
$$ |__$$ |  $$ |  $$ |  $$ |$$ |/$  \$$ |$$ |  $$ |$$ |__       $$ |
$$    $$<   $$ |  $$ |  $$ |$$ /$$$  $$ |$$ |      $$    |      $$ |
$$$$$$$  |  $$ |  $$ |  $$ |$$ $$/$$ $$ |$$ |   __ $$$$$/       $$ |
$$ |__$$ | _$$ |_ $$ \__$$ |$$$$/  $$$$ |$$ \__/  |$$ |_____    $$ |
$$    $$/ / $$   |$$    $$/ $$$/    $$$ |$$    $$/ $$       |   $$ |
$$$$$$/  $$$$$$/  $$$$$$/  $$/      $$/  $$$$$$/  $$$$$$$$/    $$/
   v1.1


[INFO]: Cleaning directory
[INFO]: Directory changed
[INFO]: command: make clean
[INFO]: command: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main.c
'
[ERROR]: Error in: cp './test_files/if/teste.x' '/home/mint/Documents/OTAWA-Studies/biowcet/din/src/main
.c' 2>&1
[ERROR]: cp: cannot stat './test_files/if/teste.x': No such file or directory
```

23

```
mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/teste.x -v -d -h
Usage: ./wcet -s archPath -p cFile [-d] [-v] [-h]
FLAGS:
        -s              Path to the architecture description file
        -p              Path to the .c file
        -d              Enable dynamic analysis
        -v              Enable verbose mode
        -h              Show this help message
EXAMPLES:
        ./wcet -s trivial -p /test_files/if/main.c
        ./wcet -s trivial -p /test_files/if/main.c -v
        ./wcet -s trivial -p /test_files/if/main.c -v -d
```

24	

mint@mint:~/Documents/OTAWA-Studies/biowcet$ ./wcet -s teste -p ./test_files/if/teste.x -j
./wcet: invalid option -- 'j'
Usage: ./wcet -s archPath -p cFile [-d] [-v] [-h]
FLAGS:
-s Path to the architecture description file
-p Path to the .c file
-d Enable dynamic analysis
-v Enable verbose mode
-h Show this help message
EXAMPLES:
./wcet -s trivial -p /test_files/if/main.c
./wcet -s trivial -p /test_files/if/main.c -v
./wcet -s trivial -p /test_files/if/main.c -v -d