UNIVERSIDADE FEDERAL DA FRONTEIRA SUL - UFFS

MODELAGEM E SIMULAÇÃO - 7ª FASE, CIÊNCIA DA COMPUTAÇÃO

Relatório de análise do desempenho do modelo de finitos serviços implementado em python 3.

JARDEL O. DUARTE

CHAPECÓ - SC
2021

1 - INTRODUÇÃO

O relatório apresenta um modelo de simulação de um caixa convencional com finitos serviços onde o número de clientes é dado na entrada após a execução do programa, contém somente um caixa de atendimento e o tempo utilizado no modelo é um intervalo uniforme de tempo (simulando unidades de minutos). Os dados a serem resultados após a experimentação são; Em relação ao atendimento, o projeto buscou contabilizar os dados da ociosidade total do caixa, a taxa de ocupação e o total de vezes em que o caixa estava livre durante a chegada de clientes; para os dados da fila foram considerados a média de espera na fila, o índice de maior ocorrência e o total de clientes que esperaram acima da média; no serviço foram levantados os dados de média do tempo trabalhado, o índice de maior ocorrência e o atendimento mais demorado do evento.

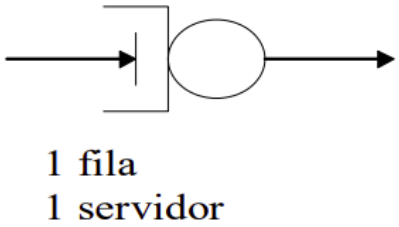
2 - DESCRIÇÃO DO PROBLEMA

O problema pode ser identificado ao considerar casos onde a ociosidade do caixa está anormal, ou a necessidade de mais processos no atendimento a serem incluídos partindo da quantidade de clientes que passam pela fila de espera durante o evento.

3 - DESENVOLVIMENTO

A simulação foi desenvolvida utilizando a linguagem de programação Python e da ferramenta Arena Simulation, a construção partiu de um centro de serviço que contém uma fila única e um servidor, esboçado na figura abaixo.

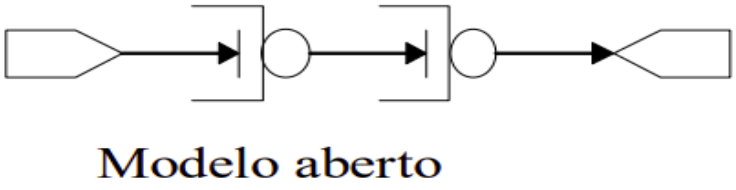
Figura 1. Centro de serviço



Fonte: Slides Modelagem e Simulação. De Mello.

O fluxo das entidades utilizando o centro de serviço é estabelecido pela chegada de um cliente na agência bancária onde pode ocorrer duas situações, sendo estas: se não houver filas, o cliente se desloca ao caixa para ser atendido pelo servidor; senão, se houver filas, o cliente espera em uma fila única até a liberação do caixa. Após os atendimentos serem finalizados os clientes se dirigem a saída onde é atualizado a disponibilidade para o cliente seguinte, a interconexão do centro de serviços pode ser representada pela figura abaixo.

Figura 2. Interconexão de modelo aberto

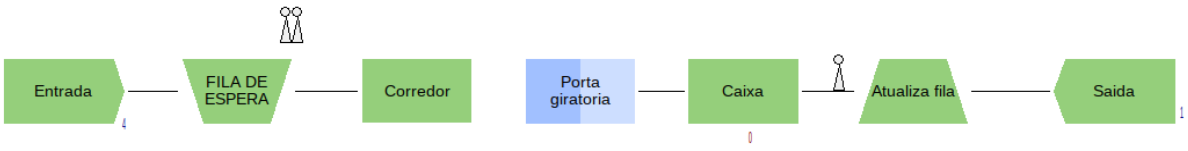


Modelo aberto

Fonte: Slides Modelagem e Simulação. De Mello.

Para uma melhor demonstrar a interconexão do centro de serviço foi implementada em um projeto no Arena Simulation com a finalidade de esboçar graficamente os fluxos definidos para o programa.

Figura 3. Fluxo de interconexão Arena



Fonte: O autor.

A construção do modelo consiste em um escalonamento por filas FIFO(First in First Out) e as variáveis utilizadas foram definidas como mostra a figura 4.

Figura 4. Variáveis do modelo

```
clientes = []
TC = []                 #TEMPO DE CHEGADA
TE = []                 #TEMPO DE ESPERA
T_IN = []               #TEMPO DE INICIO DO ATENDIMENTO
TSER = []               #TEMPO DE SERVIÇO cx1
TOCIO = []              #TEMPO OCIOSO
T_OUT = []              #TEMPO DE SAIDA
```

Fonte: O autor.

O programa foi executado em um laço de repetição *while* que continha um *if* como definição para o primeiro cliente demonstrado na figura 5.

Figura 5. Condição primeiro cliente

```
while(i < NUM_CLIENTES):
    if(i==0):   #condição para o primeiro cliente
        clientes.append(i)
        TC.append(i)
        TE.append(i)
        T_IN.append(i)
        TSER.append(rd.uniform(3, 12))
        TOCIO.append(i)
        T_OUT.append(TC[i]+TE[i]+TSER[i])
```

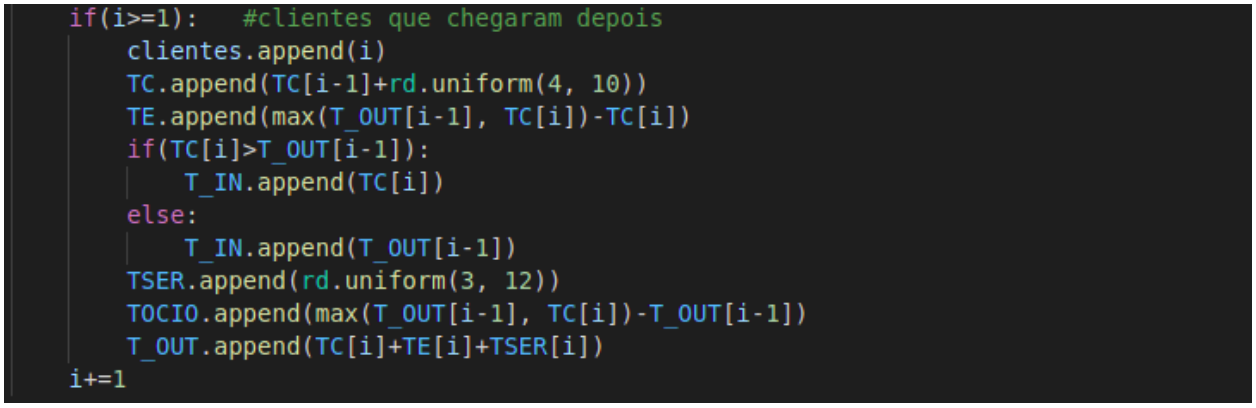
Fonte: O autor.

No primeiro instante as listas eram preenchidas da seguinte maneira:

- `clientes.append(i)`: Inclui o primeiro cliente na lista;
- `TC.append(i)`: O primeiro cliente recebe o tempo de chegada 0;
- `TE.append(i)`: tempo de espera na fila é 0;
- `T_IN.append(i)`: No mesmo instante que o cliente chega(tempo 0) o mesmo é atendido pelo operador;
- `TSER.append(rd.uniform(3, 12))`: Gera um tempo de serviço aleatório entre 3 e 12;
- `TOCIO.append(i)`: Atribui a ociosidade 0 em razão do evento começar no exato momento da chegada do primeiro cliente;
- `T_OUT.append(TC[i]+TE[i]+TSER[i])`: Define o tempo de saída somando os tempos de chegada, espere e serviço.

Para os demais clientes o programa foi implementado no *else* da condição anterior e possui as definições que estão esboçadas a seguir.

Figura 6. Condição para os demais clientes



```
if(i>=1):   #clientes que chegaram depois
    clientes.append(i)
    TC.append(TC[i-1]+rd.uniform(4, 10))
    TE.append(max(T_OUT[i-1], TC[i])-TC[i])
    if(TC[i]>T_OUT[i-1]):
        T_IN.append(TC[i])
    else:
        T_IN.append(T_OUT[i-1])
    TSER.append(rd.uniform(3, 12))
    TOCIO.append(max(T_OUT[i-1], TC[i])-T_OUT[i-1])
    T_OUT.append(TC[i]+TE[i]+TSER[i])
i+=1
```

Fonte: O autor.

Neste momento, os preenchimentos foram:

- `clientes.append(i)`: Inclui o cliente atual na lista;
- `TC.append(TC[i-1]+rd.uniform(4, 10))`: Gera um tempo de chegada aleatório maior do que a chegada anterior e no intervalo entre 4 e 10;
- `TE.append(max(T_OUT[i-1], TC[i])-TC[i])`: Tempo de espera é dado pelo decremento do tempo de saída anterior em relação ao tempo de chegada do atual;
- `T_IN.append(TC[i]) ou T_IN.append(T_OUT[i-1])`: Para atribuir os dados de início de atendimento foi utilizado uma condicional *if* | *else* para determinar se o tempo de chegada é maior que o tempo de saída anterior;
- `TSER.append(rd.uniform(3, 12))`: Gera um tempo de serviço aleatório entre 3 e 12;
- `TOCIO.append(max(T_OUT[i-1], TC[i])-T_OUT[i-1])`: O tempo de ociosidade agora passou a ser o tempo de chegada atual decrementando o tempo de saída anterior;
- `T_OUT.append(TC[i]+TE[i]+TSER[i])`: O tempo de saída se mantém sendo a soma dos tempos de chegada, espere e serviço.

Enfim, para calcular a média de espera na fila foi utilizado a fórmula $\frac{\sum_{i=0}^{n-1} TE}{n}$, e para média de tempo de serviço $\frac{\sum_{i=0}^{n-1} TSER}{n}$, sendo n o número total de clientes; O

levantamento da ociosidade do caixa foi $\sum_{n}^{n-1} TOCIO$, e a taxa de ocupação do caixa foi definida por $100 - \frac{(\sum_{i=0}^{n-1} TOCIO*100)}{TOUT[n-1]}$, sendo TOUT[n-1] a variável que define o exato instante em que o último cliente saiu do evento.

4 - RESULTADOS

Durante a execução dos testes, foram encontrados diversos resultados onde um foi escolhido para uma análise mais ampla, o teste teve como entrada dezesseis clientes e durou ≅ 1,9 milissegundos. O tempo simulado total do evento durou ≅ 109,07 minutos, que foram divididos da seguinte forma, a soma da ociosidade total do caixa foi ≅ 13,51 minutos, percentualmente 12,39% do tempo simulado, o que garantiu que o caixa estivesse livre durante oito chegadas(somando a primeira no instante em que o programa é iniciado), logo a taxa de ocupação do caixa foi de 87,61%. Para os dados da fila de espera o tempo total foi de ≅ 13,09 minutos, logo a média foi de ≅ 0,81 minutos. Os serviços somaram um total de 95,54 minutos e uma média ≅ 5,97 minutos onde o índice de maior ocorrência foi igual 7,6, o serviço mais demorado contabilizou um total de ≅ 10,87 minutos. Enfim, com intuito de melhorar as resoluções, segue abaixo a saída do programa com os dados obtidos.

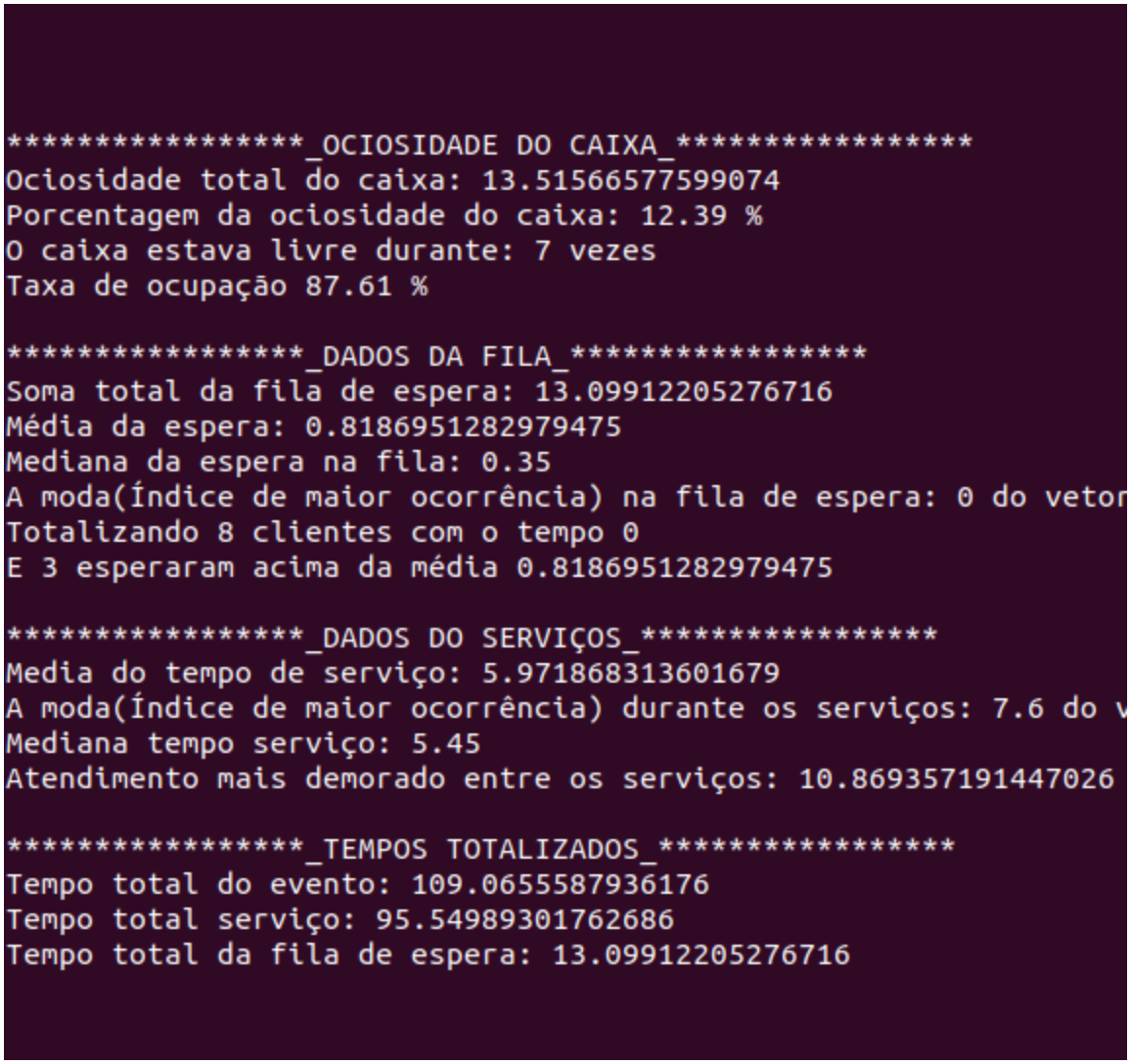
tabela 1. Tabela dos processos

Cliente	Tempo de chegada	Espera na Fila	Início do atendimento	Tempo de serviço	Ociosidade do caixa	Saída
0	0	0	0	7,64	0.0	7,64
1	6,98	0,66	7,64	6,7	0.0	14,34
2	13,58	0,76	14,34	3,92	0.0	18,26
3	20,8	0.0	20,8	4,02	2,54	24,82
4	25,53	0.0	25,53	5,17	0,72	30,7
5	34,53	0.0	34,53	6,22	3,83	40,74
6	40,01	0,74	40,74	8,64	0.0	49,39

7	47,19	2,2	49,39	5,47	0.0	54,85
8	56,8	0.0	56,8	4,91	1,94	61,71
9	60,94	0,77	61,71	4,22	0.0	65,93
10	66,86	0.0	66,86	5,43	0,93	72,29
11	71,03	1,27	72,29	10,87	0.0	83,16
12	77,28	5,88	83,16	3,47	0.0	86,63
13	86,79	0,84	86,63	5,9	0.0	92,54
14	95,35	0.0	95,35	8,88	2,82	104,23
15	104,98	0.0	104,98	4,09	0,74	109,07

Fonte: O autor.

Figura 8. Dados de saída



Fonte: O autor.

Em síntese foi observado que um dos problemas presentes no desenvolvimento se dá em razão de ser um modelo finito qual possui somente um caixa de atendimento, o que geraria problemas na capacidade da fila em uma entrada maior de clientes (em um cenário real, espaço do banco ou quantidade de cadeiras na fila espera, entre outros), por exemplo, no Arena Simulation ao considerar uma entrada uniforme de (4, 10) e um atendimento uniforme de (7, 15), a fila estourou o limite de 150 clientes quando o modelo era executado com uma entrada maior que 300 clientes o que confirma a precisão destas observações.

Por fim, a proposta de um modelo mais adequado a partir desse projeto será incluída na lista dos trabalhos futuros e espera-se que este relatório possa ser útil para

melhor definir os passos a serem elaborados.

REFERÊNCIAS

DE MELLO, Braulio Adriano. Slides Modelagem e Simulação. [S.l.]: Plataforma Moodle UFFS - Modelagem e Simulação, jun. 2021.

PARAGON. Arena Acadêmico (Student). [S.l.]: Paragon Decision Science. Disponível em: <https://www.paragon.com.br/arena-academico-student/>.