Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra
2015/16

Simulação e Computação Científica

Relatório Trabalho Final

Estudo do funcionamento de uma estação de serviço

Alexandra Tomé Leandro 2013146082
Ana Rita Cabral Dias 2013142131

Coimbra 2015/2016

1. Introdução

Este trabalho tem como objetivo o estudo do funcionamento de uma estação de serviço, para tal, foi desenvolvido um simulador interativo que representa esse estabelecimento em linguagem Java, adaptando o código que nos foi fornecido. O modelo desenvolvido irá simular os serviços de postos de abastecimento de gasolina, gasóleo e a secção da loja onde são feitos os pagamentos. A estação funciona 24 horas por dia, sendo que existem três turnos distintos.

O projeto será dividido em 3 situações diferentes, o *cenário base*, *cenário i* (uma variação simples do primeiro) e *cenário ii*.

No *cenário base*, a estação de serviço é constituída por 3 bombas, sendo 2 delas atribuídas à gasolina e a restante ao gasóleo, e uma loja onde se efetua os pagamentos depois de abastecer o veículo. Neste caso, existem 12 empregados que trabalham, estando, em cada turno, um funcionário na seção da loja e 3 divididos entre as bombas existentes. A chegada de clientes terá um intervalo de tempo que segue uma distribuição exponencial negativa de média 1.2 minutos. De todos os clientes que chegam à estação de serviço 20% pretendem abastecer gasóleo e os restantes gasolina. A operação de abastecimento não depende to tipo de combustível e demora em média 4 minutos com desvio padrão de 2.5 minutos segundo uma distribuição normal. No caso do pagamento do serviço, o tempo despendido seguirá também uma distribuição normal com média de 1 minuto e desvio de 0.5 minutos.

O *cenário i* consiste numa ligeira alteração na situação disposta acima. Neste caso, será acrescentado um novo posto idêntico aos existentes no sector da gasolina, e consequentemente implica a contratação de mais um funcionário, passando a haver no total 4 bombas.

Para o *cenário ii*, é feito um novo sistema, agora, com quatro postos self-service multifuncionais. Desta forma, o cliente pode abastecer e pagar no mesmo posto. O tempo de serviço passará a ser em média 4.5 minutos com desvio padrão de 2 minutos conforme uma distribuição normal. Para esta versão, são necessários apenas dois funcionários que supervisionam as novas máquinas.

De forma a validar o simulador construído e as suas possíveis variantes, foram efetuados vários testes, com base em critérios específicos, que permitem verificar a correção dos resultados obtidos e consequentemente a validade do trabalho efetuado.

2. Arquitetura do Simulador

De forma a desenvolver e simular os diferentes cenários e seções propostas no enunciado, o código do projeto está dividido em diferentes classes e respetivos ficheiros:

***Simulador**.java*

Esta classe servirá para implementar o cenário da simulação de todo o programa. Aqui serão criados os diferentes serviços do *cenário i* e situação no enunciado (posto de gasolina, posto gasóleo e loja) e do *cenário ii* (posto *self-service).*

É nesta classe que se encontra a função *main()* do projeto que, neste caso, carrega a interface. Na interface são definidos pelo utilizador os inputs pretendidos e é criado um simulador em conformidade. Ou seja, o construtor desta classe, que recebe todos os parâmetros da simulação, é usado na interface para a passagem dos valores definidos pelo utilizador. Assim, realiza-se a passagem dos parâmetros de cada serviço, nomeadamente o número de funcionários, a média e o desvio do tempo de serviço, bem como o tempo de simulação, a média de chegada e a escolha de cenário a utilizar.

Seguidamente é utilizado o método *executa()* do simulador definido, onde, enquanto não se atingir o tempo de simulação estipulado se procede à execução de todos os eventos do simulador, à atualização dos resultados finais e seguidamente à apresentação destes na interface.

***Servico**.java*

Na classe *Servico* são efetuadas as operações necessárias em cada serviço, nomeadamente a inserção e remoção do cliente de filas de espera, o tratamento do serviço em causa com o

seu respetivo tempo de serviço, e todas as transações envolvidas no processamento de clientes, bem como a atualização dos resultados medidos e o cálculo final destes.

***ListaEventos**.java*

Como o nome indica, esta classe conterá uma lista dos eventos atuais do simulador. Assim, procede-se, com o auxílio de métodos da classe Evento, à inserção de eventos na lista desta classe na posição correta, ou seja, por ordem crescente dos instantes de ocorrência dos eventos.

***Evento**.java*

Esta classe abstrata funciona com uma classe auxiliar utilizada pelo simulador de modo a poderem ser iniciados diferentes eventos dentro do simulador.

***Chegada**.java*

Esta classe é uma subclasse de Evento e como tal, tem acesso ao construtor da superclasse. Será simulada a chegada de clientes à estação de serviço. Um novo cliente irá ser gerado através do construtor da classe *Cliente* e, através do tipo de cliente gerado (definido na classe *Cliente*) e do tipo de cenário escolhido pelo utilizador, proceder-se-á ao encaminhamento do cliente para o serviço correto. Posteriormente, será agendado o evento da chegada de novo cliente de acordo com o tempo calculado pela distribuição exponencial na classe *Aleatorio*.

***Saida**.java*

É também uma subclasse da classe Evento. No entanto, no seu método *executa*() trata a saída de um cliente de um determinado serviço, isto é, um cliente será retirado do serviço em questão de modo a dar seguimento ao seu percurso na estação de serviço. No caso do simulador presente, esta classe irá retirar o cliente do serviço loja depois de este efetuar o pagamento, ou retirá-lo do serviço self-service depois de ter sido completado o serviço, dependendo do cenário em questão, e processar a saída do cliente da simulação, ou retirar o cliente.

***Cliente**.java*

Aqui será simulado cada cliente, sendo definido de forma aleatória o tipo de cliente: se pretende abastecer o veículo com gasolina ou de gasóleo.

***Transfere**.java*

Esta subclasse da classe Evento não será utilizada no *cenário ii. P*ermite fazer a transição dos clientes, ou seja, transferir o cliente para o serviço de pagamento na loja e retirá-lo do serviço atual (gasolina ou gasóleo). Para isso, utiliza-se o método de remoção de serviços da classe *Servico*. Esse método irá devolver o cliente que foi removido, sendo este posteriormente colocado na loja através desta classe. O evento da transição é colocado no método da classe Serviço especificado.

***Aleatorio**.java*

Aqui são definidas as funções que geram os tempos n*ecessários* ao simulador de acordo com função exponencial (com média) ou função normal (com média e desvio padrão).

A função exponencial é aplicada no agendamento de chegadas de clientes, na classe *Chegada* e a função normal é utilizada nas inserções e remoções de clientes nos serviços.

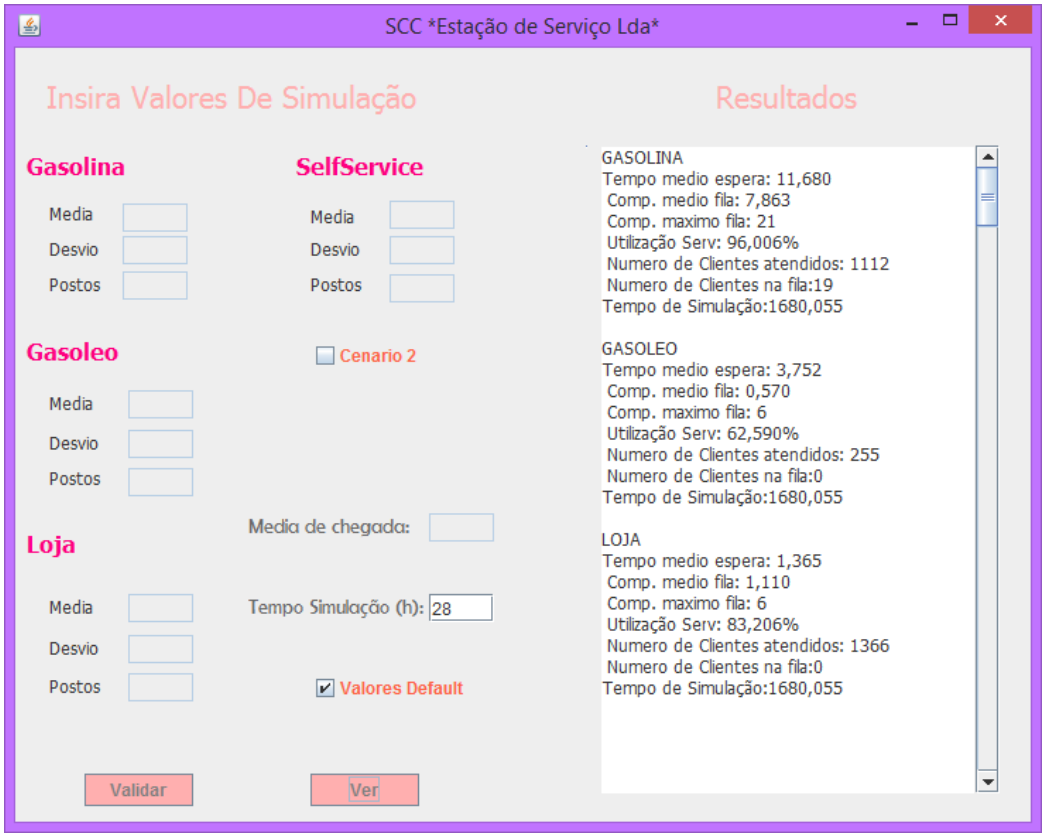
Deve-se salientar que na função normal descartam-se os valores negativos característicos da distribuição normal, no entanto, para não se proceder à perda de tantos valores podíamos ter recorrido à utilização do módulo para garantir que os tempos calculados eram sempre positivos. Ainda, esta função devolve um vetor com 2 valores, uma vez que calcula tempos aos pares. Estes são guardados na classe *Servico* e a função só é chamada quando necessário, quando não existe nenhum novo valor calculado.

***RadomGenerator**.java*

Esta classe é apenas um auxiliar da classe *Cliente* e *Aleatorio* que permite obter uma sequência de números pseudo-aleatórios necessários para os cálculos das funções normais e exponenciais. Através da utilização de diferentes *seeds* devidamente espaçadas para diferentes conjuntos de números calculados é possível obter bons resultados. Tem os métodos *rand()* e

rand64() que devolvem os dois número aleatório de acordo com a *seed* passada como parâmetro.

Interface



A interface é usada para o utilizador inserir os parâmetros e visualizar os resultados. Como se pode ver, nela podemos escolher usar o cenário 2, ou seja, o *cenário ii* do posto self-service. Caso não especifiquemos através do *check* que queremos utilizar cenário 2 é usado o cenário 1, que neste caso possuí 3 serviços separados (gasolina, gasóleo e loja). Repare-se que só podemos inserir valores nas caixas dos parâmetros respetivos do cenário escolhido.

Ainda, podemos inserir valores desejados ou escolher, através de um *check,* utilizar valores *default* e assim apenas teremos de especificar tempo de simulação. Os valores *default* neste caso são os parâmetros especificados no enunciado do *cenário i* para o cenário 1 e do *cenário ii* caso esteja selecionado o cenário 2.

Os tempos de serviço e chegada pedidos ao utilizador terão de ser inseridos em minutos, no caso do tempo de simulação, para facilitar, devido à necessidade de simular várias horas o parâmetro inserido será em horas. No caso dos resultados apresentados, as unidades presentes dos tempos estão também em minutos, unidade com a qual o simulador trabalha.

Abordagem da Simulação

Para facilitar o estudo dos diferentes cenários alternativos o simulador foi feito de modo a ser adaptável, assim, é possível escolher um de dois cenários: um modelo de três seções (gasolina, gasóleo e loja), e um modelo self-service de postos multifuncionais. Além disto, é ainda possível modificar os tempos de chegada, de serviço e o número de postos existentes. Deste modo o código elaborado é o mais geral possível, permitindo uma fácil especificação dos dados. A visualização dos resultados por recurso de interface gráfica é bastante fácil e intuitiva.

Com o objetivo de calcular os tempos de serviço que seguem uma distribuição normal, foi acrescentado ao código disponibilizado na classe *Aleatorio* o método *Normal()*que receberá como parâmetros a média e o desvio padrão do tempo de serviço desejado de forma a devolver

um vetor de números pseudo-aleatórios que irão ser utilizados pelo simulador. Esta classe é auxiliada pela classe fornecida *RandomGenerator* na geração de números de acordo com a *seed* do serviço em questão, o que permite criar variáveis aleatórias distintas que condicionam os resultados, tal como aconteceria na vida real.

No código inicialmente fornecido, existia apenas um serviço indistinto. No entanto, com a existência de diferentes serviços, no caso do *cenário base* ou *cenário i*, foi necessário criar uma classe que tratasse de encaminhar os clientes para o serviço seguinte, a classe *Transfere*, que como vimos anteriormente, encarrega-se da saída do cliente de um certo serviço e da sua entrada noutro serviço, que será sempre a loja para efetuar pagamento. Esta classe não é necessária no *cenário ii*, pois existe apenas um serviço, o self-service.

A classe *Servico* foi também adaptada, de modo a utilizar a classe anterior, assim, nos cenários de três seções distintas, se o cliente se encontra no serviço da gasolina ou gasóleo será direcionado para a loja através da classe *Transfere,* se o cliente estiver já no serviço da loja, então irá ser utilizada a classe *Saida* que processa a saída da simulação do cliente em causa. O *Servico* foi também modificado de modo a poder ter diversos postos, assim, o serviço só será considerado ocupado se nenhum dos postos estiver livre. Ainda, nesta classe, nos valores estatísticos do serviço, foi acrescentado o comprimento máximo da fila de espera, e a taxa de utilização do serviço foi alterada de modo a ter em conta o número de postos que o serviço presente contém.

Os diferentes serviços necessários são criados na classe *Simulador*, desta forma, dependendo do cenário serão criados três serviços distintos: gasolina, gasóleo e loja, ou um serviço denominado self-service. O simulador terá como critério de paragem o tempo de simulação escolhido pelo utilizador na interface, e será condicionado pelos parâmetros escolhidos para os tempos de chegada, serviço e número de postos. Estes valores escolhidos serão utilizados para inicializar os serviços respetivos. Assim, a classe está adaptada para os diferentes cenários, e vai retirar as informações relativas a cada serviço e imprime-as na interface.

Com o objetivo de responder à alínea d), relativa à rentabilidade dos cenários, foram criados na classe *Simulador* os métodos *getReceitas()*, *getDespesas()* e *getLucro(),* que calculam as receitas, os custos salariais e o lucro da estação de serviço, respetivamente. Neste caso, é tido em conta o número de funcionários e o tempo da simulação para calcular o valor total dos salários pagos. Para as receitas é utilizado o número de clientes que foram atendidos e o valor que cada cliente irá amortizar no investimento. Por fim, considerando o investimento inicial de cada cenário, é calculado o balanço entre as despesas e receitas da estação. Repare-se que estas funções são auxiliares ao nosso cálculo, que efectuamos de mês a mês, e foram pensadas para tempos de simulação correspondentes, não funcionando para outros casos (por exemplo 2 meses e 1h).

3. Validação do Simulador

Na validação de um simulador, pretende-se assegurar que o modelo presente é válido, ou seja, que representa de forma adequada o sistema real pretendido. A validação de um simulador deste tipo passará por vários critérios, nomeadamente: comparação com outros simuladores já validados e validação interna por determinação de tempo estável mínimo e análise por parâmetros.

Assim, para determinar o tempo mais próximo da "experiência ideal", ou seja, o tempo mínimo de simulação que produz resultados estáveis deve-se partir sempre do simulador vazio e parado e analisar as taxas de utilização dependentes do tempo de simulação. O valor de tempo de simulação que fixarmos como estável nesta fase será o valor usado nas seguintes etapas.

Ainda, irá ser realizada a análise dos parâmetros, que permitirá retirar conclusões do simulador modificando cada parâmetro separadamente. Nesta fase devem-se formular hipóteses lógicas (por exemplo, sabemos que, se a média de chegada de clientes aumentar, irá ser gerada uma menor quantidade de clientes no mesmo tempo de simulação) e concluir se o simulador produz resultados de acordo com essa hipótese de forma a validar o nosso simulador.

Por fim, será efetuada a comparação dos resultados obtidos no simulador Java desenvolvido com outros simuladores. Neste caso será feita com um modelo GPSS, já que este também usa primitivas de alto nível e é rápido e simples.

3.1 Validação interna

Este método funcionará como um critério de estabilização, onde se varia sucessivamente a duração da experiência nas condições inicias, analisando os resultados produzidos até à sua estabilização.

3.1.1 Validação interna *Cenário i*

Uma vez que procedemos à simulação de uma estação de serviço de 24h não faria sentido, para este modelo, simular por exemplo apenas 1h de duração da experiência. Por isso, realizámos medições com tempos de simulação que rondam as 24 horas. Na tabela abaixo podemos ver as taxas de utilização dos diferentes serviços em função do tempo de simulação, elaboramos também um gráfico com os valores da tabela, para auxiliar na fixação do tempo de simulação.

Horas	Taxa de utilização		
	Gasolina	Gasóleo	Loja
15	94,90%	64,20%	84,90%
16	95,20%	65,60%	85,80%
17	95,50%	64,40%	85,90%
18	95,80%	64%	86,30%
19	95,90%	63,90%	85,70%
20	96,20%	64,20%	85,80%
21	96,40%	64,50%	86,50%
22	96,10%	64,40%	86,70%
23	96,20%	63,40%	86%
24	96,40%	63%	86%
25	96,10%	62,80%	85,60%
26	**96,20%**	**62,50%**	**85,70%**
27	96,10%	63,60%	85,80%
28	96,20%	63,70%	85,80%
29	96,40%	63,90%	85,80%
30	96,50%	65,80%	85,90%

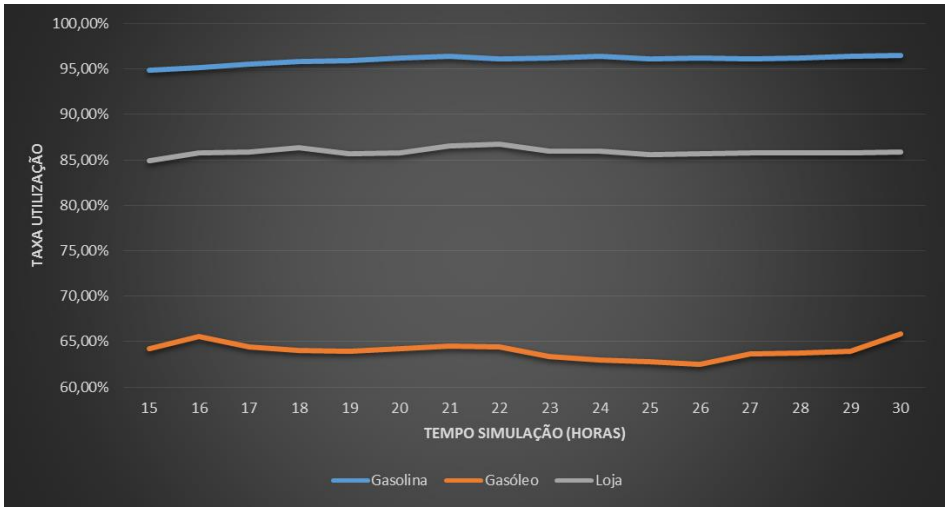


Figura 1 - Taxas de Utilização dos serviços em função do tempo de simulação

Pode-se verificar que o simulador começa a estabilizar por volta das 25h de simulação. Foi considerado o tempo fixo de estabilização de 26h para este cenário, uma vez, que apresenta melhores resultados gerais.

3.1.2 Validação interna *Cenário ii*

Mais uma vez foram efetuadas medições com tempos adequados ao problema que se está a simular. Apresentamos a tabela da taxa de utilização do serviço self-service consoante as horas de simulação e o respetivo gráfico.

Horas	Taxa utilização
15	93,5%
16	93,9%
17	94,3%
18	94,6%
19	94,9%
20	95,1%
21	94,8%
22	94,8%
23	94,3%
24	94,5%
25	94,4%
26	**94,1%**
27	94,1%
28	94,3%
29	94,4%
30	94,6%

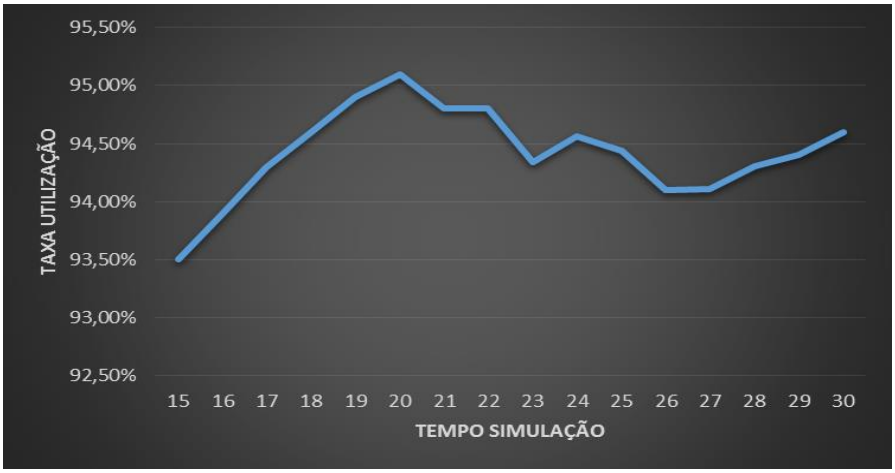


Figura 2 -Taxas de Utilização do serviço em função do tempo de simulação

Neste gráfico os valores parecem muito discrepantes devido à escala usada, no entanto a maioria dos valores ronda os 94%. Foi escolhido como tempo fixo de estabilização as 26h uma vez que, a partir deste tempo, ocorre estabilização.

3.2 Análise de parâmetros

Esta validação é efetuada através de testes de sensibilidade, que demonstram a forma como os efeitos de alterações nos parâmetros do modelo vão corresponder a modificações do sistema real. É formulada uma hipótese de performance do sistema para cada valor a alterar, os resultados obtidos são depois analisados e comparados para tirar conclusões. Repare-se que, entende-se por valores *default* os valores que são indicados no enunciado para cada cenário, e todos os resultados de testes efetuados são comparados com os resultados dos valores *default* apresentados inicialmente. Ainda, os parâmetros são mudados de forma separada para a avaliação dos resultados nas mesmas condições.

3.2.1 Análise de parâmetros *cenário i*

Abaixo apresentamos a tabela com os resultados da simulação nas condições *default*: 4 bombas (1 de gasóleo e 3 de gasolina), e um posto na loja, tempos de serviço de abastecimento que seguem *N(4, 2.5)* e de pagamento de *N(1, 0.5),* com intervalos de tempo de chegadas segundo *E(1.2)*. O tempo de simulação utilizado foi o tempo fixo calculado anteriormente para este cenário, 26 horas:

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,084	8,024	25	96,212%	1034	2
Gasóleo	4,071	0,621	5	62,541%	238	0
Loja	1,783	1,452	7	85,687%	1271	0

a) Variação do tempo de simulação

Neste teste iremos comparar tempos de simulação menores e maiores que o tempo de simulação menor estável escolhido para o cenário que é 26 horas.

Hipótese: Neste caso, como apenas existe variação de 1 ou 2 horas em relação ao tempo fixo escolhido, espera-se que não se verifiquem grandes alterações de resultados. No entanto, tempos de simulação menores que o tempo fixado de estabilização poderão apresentar resultados mais discrepantes que as restantes simulações, pois estarão abaixo do tempo mínimo de estabilidade determinado.

Tabela de tempo de simulação superior (28 horas):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	11,837	7,926	25	96,246%	1115	10
Gasóleo	4,206	0,651	5	63,670%	257	3
Loja	1,77	1,448	7	85,792%	1370	4

Tabela de tempo de simulação superior (27 horas):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	11,839	7,870	25	96,127%	1077	0
Gasóleo	4,257	0,654	5	63,606%	249	0
Loja	1,466	1,466	7	85,82%	1321	4

Tabela de tempo de simulação inferior (25 horas):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,138	8,124	25	96,060%	995	9
Gasóleo	4,228	0,643	5	62,751%	228	0
Loja	1,799	1,465	7	85,621%	1221	1

Tabela de tempo de simulação superior (24 horas):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,504	8,371	25	96,384%	959	5
Gasóleo	4,213	0,641	5	63,048%	219	0
Loja	1,8332	1,497	7	85,961%	1173	4

Podemos comprovar que os resultados obtidos estão de acordo com a hipótese formulada à *priori*.

b) Variação do tempo médio de chegada

Neste caso iremos simular com médias de chegada maiores e menores que a *default* usada e comparar os resultados.

Média Default=1,2 minutos

Média menor=0,2 minutos

Média maior=2,2 minutos

Hipótese: A média de chegada controla a quantidade de clientes que chega ao simulador, já que define o tempo entre chegadas. Quanto maior for a média de chegada, menos clientes são gerados, uma vez que em média o tempo entre geração de clientes irá aumentar. De forma análoga, quanto menor a média de chegada maior será a afluência de clientes. Assim, esperam-se resultados em conformidade.

Tabela de média de Chegada maior (2,2 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	0,620	0,227	4	52,920%	570	0
Gasóleo	1,649	0,137	2	35,118%	130	0
Loja	0,543	0,243	4	47,480%	699	0

Tabela de média de chegada menor (0,2 minutos):

	Tempo médio espera	Comp. médio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	639,674	2601,275	5265	99,978%	1079	5266
Gasóleo	579,376	560,332	1149	99,819%	359	1150
Loja	5,847	5,385	17	96,203%	1435	2

Como se pode verificar os resultados estão de acordo com a hipótese. Quando se diminuí a média de chegada de clientes (0,2 minutos), verifica-se um aumento geral do comprimento médio da fila dos serviços, dos clientes que ficaram na fila, do tempo médio de espera e dos clientes atendidos, como esperado, porque a quantidade de clientes nos serviços é maior e os outros parâmetros (postos e médias de tempo de serviço) se mantêm.

Pelo contrário, quando se aumenta a média de chegada (2,2 minutos), os clientes irão aparecer com mais tempo de intervalo entre eles, e por isso verifica-se o contrário da situação anterior. Para os mesmos parâmetros restantes observa-se uma diminuição geral do comprimento médio da fila dos serviços, dos clientes que ficaram na fila e do tempo médio de espera. Repare-se ainda que as percentagens de utilização do serviço diminuem porque existem menos clientes.

Ainda, quando existem mais clientes verificam-se mudanças mais drásticas nos serviços de combustível do que no de pagamento, o que é normal porque a maioria dos clientes ficará retido antes de chegar à loja.

c) Variação do tempo médio de serviço

1. Gasolina

Média Default= 4 minutos

Média maior= 5 minutos

Média menor= 3 minutos

Hipótese: A média da distribuição normal da gasolina diz respeito diretamente ao tempo de serviço do serviço gasolina. Assim, espera-se que os resultados do gasóleo não sejam muito afetados.

Tabela de Média maior (5 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	108,070	71,747	143	99,761%	902	134
Gasóleo	4,071	0,621	5	62,547%	238	0
Loja	1,017	0,74	5	77,012%	1139	0

Tabela de Média menor (3 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	2,300	1,530	12	79,397%	1038	0
Gasóleo	4,071	0,621	5	62,538%	238	0
Loja	3,411	2,788	15	85,981%	1274	1

Os resultados estão de acordo com a hipótese. Para médias maiores, o tempo de atendimento, comprimento das filas e utilização do serviço de gasolina aumentam porque os clientes demoram mais tempo a ser atendidos e por isso o serviço fica mais lento. Assim, ficam também mais clientes na fila de espera, o que leva à diminuição de utilização da loja, uma vez que receberá menos clientes da gasolina.

Caso se diminua a média passa-se exatamente o contrário. O serviço de gasolina fica mais ágil e aumentam o número de clientes atendidos na gasolina e, consequentemente, na loja.

2. Gasóleo

Média default= 4 minutos

Média maior= 5 minutos

Média menor= 3 minutos

Hipótese: Analogamente, espera-se que os tempos de serviço do gasóleo sejam afetados. Quanto maior a média, maior será tempo de espera e quanto menor média, menor será tempo de espera.

Tabela de Média maior (5 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,084	8,023	25	96,212%	1034	2
Gasóleo	7,763	1,189	7	77,584%	237	0
Loja	1,770	1,443	7	85,681%	1271	1

Tabela de Média menor (3 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,085	8,022	25	96,213%	1035	1
Gasóleo	2,918	0,445	4	51,734%	239	1
Loja	1,784	1,453	7	85,652%	1270	1

De acordo com o esperado, o serviço gasóleo fica mais rápido, atende mais clientes e tem menor fila de espera quando as suas médias são menores para os restantes parâmetros constantes. Pelo, contrário, quando a média é maior, o atendimento demora mais tempo e é menos eficiente. Os resultados do serviço da loja variam de forma semelhante ao teste anterior da gasolina.

3. Loja

Média default= 1 minutos

Médias maiores: 2 e 1,5 minutos

Média menor:0,5 minutos

Hipótese: Uma vez que a média diz respeito ao tempo que a loja demora a processar os clientes, espera-se que: os resultados do gasóleo e gasolina não sejam afetados e que caso: 1) média seja maior para os mesmos parâmetros restantes, o número de clientes que concluí pagamento diminua e as filas/tempos de espera da loja aumentem; 2) a média seja menor, os clientes que se dirigem para a loja concluam o pagamento com maior rapidez e menos tempo/tamanho de fila.

Tabela de Média maior (1,5 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,085	8,022	25	96,213%	1035	1
Gasóleo	4,071	0,621	5	62,551%	238	0
Loja	152,475	124,276	257	991551%	1014	258

Tabela de Média maior (2 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,084	8,023	25	96,213%	1034	2
Gasóleo	4,071	0,621	5	62,547%	238	0
Loja	303,865	27,498	504	99,615%	767	504

Tabela de Média menor (0,5 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,084	8,023	25	96,212%	1034	2
Gasóleo	4,071	0,621	5	62,544%	238	0
Loja	0,298	0,243	3	53,906%	1272	0

Os resultados estão de acordo com a hipótese, podemos reparar que para médias maiores, o serviço da loja fica mais lento e cada vez menos clientes são atendidos porque cada cliente demora mais tempo. No entanto, para médias menores são atendidos mais clientes de forma mais rápida, ou seja, a taxa de utilização do serviço diminuí.

d) Variação do desvio padrão do tempo de serviço

Hipótese: Não se podem tirar muitas conclusões através do desvio. Quanto maior for, mais distante da média os resultados poderão estar, e quanto menor, menos afastado da média os valores serão. No entanto, existe, normalmente, maior percentagem de resultados próximos à média numa distribuição normal, portanto resultados podem não ser muito expressivos.

Note-se que, neste caso, a média diz respeito ao tempo de atendimento do serviço do qual se vai variar o desvio. Assim, quanto maior o desvio, maior deverá ser o tempo de atendimento do serviço em que se aumenta o desvio. Pelo contrário quanto menor desvio, menor deverá ser tempo de serviço do serviço.

Os serviços onde não se mudam desvios devem manter resultados semelhantes, a não ser que, no caso do gasóleo ou gasolina, as diferenças sejam suficientemente expressivas para modificar o número de clientes que chegam à loja (mudando os seus resultados).

1. Gasolina

Desvio default=2.5 minutos

Desvio maior=2.6 minutos

Desvio menor=2.4 minutos

Tabela de Desvio maior (2,6 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	13,335	8,852	26	96,910%	1034	2
Gasóleo	4,071	0,621	5	62,550%	238	0
Loja	1,831	1,491	7	85,666%	1271	0

Tabela de Desvio menor (2,4 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	10,636	7,063	24	95,263%	1033	3
Gasóleo	4,071	0,621	5	62,537%	238	0
Loja	1,855	1,510	9	85,601%	1270	0

Dado que apenas aproveitamos valores positivos, caso a média seja baixa, ter desvio elevado pode aumentar tempo de serviço. Caso desvio seja pequeno, garantimos que valores estão mais próximos da média. Neste caso, as diferenças são quase impercetíveis, talvez porque a variação é de apenas 0.1 minutos, mas em geral, maior desvio conduz a maiores tempos de espera e maior fila na gasolina. Verifica-se que resultados concordam com a hipótese.

2. Gasóleo

Desvio default=2.5 minutos

Desvio maior=2.6 minutos

Desvio menor=2.4 minutos

Tabela de Desvio maior (2,6 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,085	8,022	25	96,213%	1035	1
Gasóleo	4,433	0,676	5	64,274%	237	1
Loja	1,755	1,429	6	85,568%	1269	2

Tabela de Desvio menor (2,4 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,084	8,023	25	96,212%	1034	2
Gasóleo	3,787	0,580	5	62,214%	239	0
Loja	1,756	1,432	7	85,682%	1271	1

De forma semelhante à gasolina existe pouca variação de resultados, no entanto, desvio maior torna gasóleo mais lento e desvio menor torna gasóleo mais ágil. Confirma-se que resultados estão de acordo com hipótese.

3. Loja

Desvio default=0,5 minutos

Desvio maior=0,6 minutos

Desvio menor=0,4 minutos

Tabela de Desvio maior (0,6 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,085	8,022	25	96,213%	1035	1
Gasóleo	4,071	0,621	5	62,551%	238	0
Loja	2,509	2,045	10	89,051%	1269	3

Tabela de Desvio menor (0,4 minutos):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,083	8,024	25	96,211%	1034	2
Gasóleo	4,071	0,621	5	62,537%	238	0
Loja	1,561	1,272	9	83,744%	1271	0

Mais uma vez, não há muito a concluir com a variação do desvio padrão da Loja, no entanto, os resultados são semelhantes aos verificados com a variação do desvio padrão da gasolina e do gasóleo e podemos concluir que os resultados estão em concordância com a hipótese.

e) Variação do número de postos

Hipótese: Quanto maior for o número de postos de um serviço mais rápido será esse serviço, ou seja, os tempos de espera bem como o comprimento das filas diminuirão e mais clientes poderão ser atendidos. Quanto menor for o número de postos de determinado serviço menos ágil será esse serviço, ou seja as médias de espera e filas aumentarão e haverá menos capacidade para escoar clientes. Isto para valores estáticos da média de tempo de serviço e desvio.

No caso da loja só os resultados da loja são afetados com a mudança de postos. Nos serviços de combustível, quando se aumenta número de postos, mais clientes chegam à loja (aumentam clientes atendidos no combustível) e, quando se diminuem o número de postos, o número de clientes que chegam à loja diminuí, traduzindo-se essa situação nos resultados.

1. Gasolina

Postos default=3;

Tabela de maior número de postos (4):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	1,082	0,719	7	72,294%	1037	0
Gasóleo	4,071	0,621	5	62,537%	238	0
Loja	3,286	2,684	15	85,984%	1274	0

Tabela de menor número de postos (2):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	248,808	165.288	323	99,881%	713	324
Gasóleo	4,053	0,621	5	62,559%	239	0
Loja	0,559	0,341	3	64,213%	951	0

2. Gasóleo

Neste caso, o número de postos *default* é 1, sendo assim, não faz sentido medir resultados com valores abaixo, porque o número de postos é sempre maior ou igual a 1.

Tabela de maior número de postos (2):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,084	8,024	25	96,212%	1034	2
Gasóleo	0,298	0,046	2	31,295%	239	0
Loja	2,034	1,658	8	85,687%	1271	1

3. Loja

Neste serviço, o número de postos *default* também tem valor igual a 1 e por isso, também não simularemos com valores abaixo do *default*.

Tabela de maior número de postos (2):

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,083	8,024	25	96,211%	1034	2
Gasóleo	4,071	0,621	5	62,537%	238	0
Loja	0,078	0,064	2	42,878%	1271	0

Podemos verificar que os resultados em todos os serviços estão de acordo com a nossa hipótese. Assim, quanto mais postos tiver o serviço, mais ágil fica, e quanto menos postos tiver mais lento fica (com maior fila e tempos de espera). Salienta-se que, o aumento do número de postos, tem mais expressão e é mais necessário em serviços que tenham mais quantidade de clientes, como é o caso da gasolina e Loja. No gasóleo também se obtém maior agilidade, mas este aumento de postos é um pouco desnecessário do ponto de vista do custo/benefício, já que o serviço não era lento antes.

3.2.2 Análise de parâmetros *Cenário ii*

Tabela de resultados com valores *default: 4 postos multifuncionais self-service* onde o tempo de serviço segue *N(4.5, 2)* e o intervalo de chegadas dos clientes será também *E(1.2).*

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
7,283	5,952	23	94,175%	1276	0

a) Variação do tempo de simulação

Hipótese: Uma vez que o tempo de simulação escolhido antes, é o tempo mínimo de estabilidade (26 horas), espera-se que tempos maiores que este apresentem resultados mais semelhantes a este do que tempos menores.

Tabela de resultados com tempo de simulação 27 horas:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
7,120	5,818	23	94,113%	1324	0

Tabela de resultados com tempo de simulação 25 horas:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
7,476	6,135	23	94,436%	1225	6

Os resultados estão em concordância com a hipótese.

b) Variação da média de chegada

Hipótese: Analogamente ao cenário i, espera-se que, com o aumento da média de chegada, o número de clientes atendidos e as filas de espera/tempos de atendimentos diminuam uma vez que chegam ao serviço clientes em intervalo de tempo maiores. Com a diminuição da média espera-se, pelo contrário, que o número de clientes aumente e por isso aumente o número de clientes atendidos e que o serviço fique mais lento (filas maiores/mais tempo de serviço).

Tabela de resultados com média de chegada 2,2 minutos:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
0,463	0,208	6	51,735%	700	0

Tabela de resultados com média de chegada 0,2 minutos:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
637,739	3210,180	6501	99,969%	1352	6502

Os resultados estão de acordo com a hipótese.

c) Variação do tempo médio de serviço

Hipótese: Analogamente ao cenário i, com aumento da média de serviço espera-se que o serviço demore mais tempo a ser concluído em cada cliente e que por isso, o número de clientes atendidos diminua e o serviço se comporte de forma lenta. Com a diminuição da média do serviço, o tempo de atendimento de cada cliente diminuiu e por isso o comportamento do sistema será exatamente o contrário.

Tabela de resultados com média de serviço 5,5 minutos:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
103,872	84,827	162	99,652%	1114	160

Tabela de resultados com média de serviço 3,5 minutos:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
1,383	1,131	14	76,116%	1276	0

Os resultados estão de acordo com a hipótese.

d) Variação do tempo médio de serviço

Hipótese: Analogamente ao cenário i, não se podem tirar conclusões através da variação do desvio padrão separadamente, já que apenas corresponde à variação dos resultados em relação à média (maiores desvios podem gerar valores mais afastados da média e menores geram valores mais próximos à média) e os resultados concentram-se mais próximos da média nas distribuições normais.

Tabela de resultados com desvio padrão 2,1 minutos:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
7,795	6,365	23	94,534%	1275	0

Tabela de resultados com desvio padrão 1,9 minutos:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
6,656	5,431	23	93,740%	1274	0

Não há muito a concluir a partir do desvio padrão, no entanto geralmente maiores desvios dão origem a serviços mais lentos do que desvios maiores.

e) Variação do número de postos de atendimento

Hipótese: Quando o número de postos aumenta o serviço fica mais ágil e com menos fila.

Tabela de resultados com 5 postos:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
1,126	0,920	11	75,340%	1276	0

Tabela de resultados com 3 postos:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
170,832	**139,593**	**267**	**99,810%**	**1009**	**266**

Os resultados estão de acordo com a hipótese.

3.3 Comparação com outro simulador

Este critério implica a comparação dos resultados de simulação com outro simulador que sabemos válido, foi utilizado um modelo de GPSS, uma linguagem de programação de simulações de tempo discreto fácil de usar. Foi, portanto, criado uma implementação deste simulador em GPSS para verificar se os resultados do simulador Java criado estão dentro dos valores aceitáveis.

De modo a validar os dois cenários principais desenvolvidos (postos self-service, e três seções distintas para loja, gasóleo e gasolina), criámos dois modelos equivalentes a cada cenário em GPSS. Assim, obtemos uma versão fidedigna do simulador a estudar que pode ser utilizada como termo de comparação para validar o simulador que foi desenvolvido.

3.3.1 Comparação com GPSS - *Cenário i:*

```
GASOLINA    STORAGE 3
GASOLEO     STORAGE 1
LOJA        STORAGE 1

            GENERATE (Exponential(1,0,1.2))
            TRANSFER  0.2,,lGASOLEO
lGASOLINA   QUEUE fila
            ENTER GASOLINA
            DEPART fila
            ADVANCE (ABS(Normal(3,4,2.5)))
            LEAVE GASOLINA
            TRANSFER  ,lLOJA
lGASOLEO    QUEUE fila1
            ENTER GASOLEO
            DEPART fila1
            ADVANCE (ABS(Normal(4,4,2.5)))
            LEAVE GASOLEO
lLOJA       QUEUE fila2       ;loja
            ENTER LOJA
            DEPART fila2
            ADVANCE (ABS(Normal(5,1,0.5)))
            LEAVE LOJA
            TERMINATE

GENERATE 1560
TERMINATE 1
```

Figure 3 - Código do modelo GPSS equivalente ao cenário i

Neste modelo GPSS começamos por definir a STORAGE das três seções presentes: gasolina, gasóleo e loja, esta definição corresponde ao número de postos/funcionários disponíveis para atendimento em cada seção da estação de serviço.

De seguida, é simulada a chegada de clientes através da função Exponential de média 1.2 minutos. Os clientes são separados distribuindo-se entre gasolina e gasóleo numa relação de 80% e 20%, respetivamente, isto é feito pelo comando TRANSFER que direcionará clientes específicos para a gasolina ou gasóleo.

Dentro da seção respetiva lGASOLINA e lGASOLEO, o cliente irá ficar numa fila de espera até ser atendido, o tempo de serviço é determinado através da função Normal, que calculará os tempos de acordo com uma distribuição normal de média 4 minutos e desvio 2.5 minutos. Repare-se que aqui usamos o módulo para garantir que os valores da distribuição são positivos, em oposição ao descarte dos valores negativos que utilizámos no simulador JAVA.

Posteriormente, o cliente segue para a última seção que corresponde à lLOJA onde irá novamente entrar numa fila de espera até ser atendido, o cliente é processado durante um tempo obtido por distribuição normal de média 1 e desvio 0.5 minuto. Finalmente, o cliente sai da simulação com o comando TERMINATE.

Este processo ocorrerá pelo tempo definido, em minutos, pela linha GENERATE 1560.

Aqui apresentamos a comparação dos resultados obtidos no modelo GPSS e no simulador Java, para o mesmo tempo de simulação, 26 horas:

		Tempo médio espera	Comp. Med. fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos
Java	**Gasolina**	12,084	8,024	25	96,212%	1034
	Gasóleo	4,071	0,621	5	62,541%	238
	Loja	1,783	1,452	7	85,687%	1271
GPSS	**Gasolina**	5,020	3,131	16	85,7%	969
	Gasóleo	4.002	0,603	5	63,5%	235
	Loja	1,281	0,986	8	76,3%	1198

O GPSS apresenta valores semelhantes ao simulador JAVA, as variações devem-se provavelmente ao uso diferente das distribuições normais, uso do módulo no gpss e do descarte no JAVA. Podemos observar que os resultados de ambos modelos são consistentes, sendo que as taxas de utilização de cada seção apresentam diferenças entre 10% e 1% nos diferentes simuladores. Além disto, vemos também que, a gasolina apresenta nos dois modelos maior tempo médio de espera e maior comprimento médio e máximo da fila de espera. Podemos comparar ainda, o número de clientes atendidos em cada área dos simuladores e verificar a sua proximidade.

Assim, podemos concluir que o modelo desenvolvido em Java apresenta valores corretos e portanto, deverá ser considerado válido.

3.3.2 Comparação com GPSS - *Cenário ii*:

```
SELF    STORAGE 4
        GENERATE (Exponential(1,0,1.2))
        QUEUE fila
        ENTER SELF
        DEPART fila
        ADVANCE (ABS(Normal(2,4.5,2)))
        LEAVE SELF
        TERMINATE

        GENERATE 1560
        TERMINATE 1
```

Figure 4 -Código do modelo GPSS equivalente ao cenário ii

Neste cenário, o modelo GPSS será bastante mais simples que o anterior. Como os postos são self-service, não é necessária distinção entre as seções vistas antes, assim, é definido na STORAGE o número de postos existentes. Os clientes são gerados de modo análogo pela função Exponential, de seguida entram na fila de espera em QUEUE fila. Quando existir um posto disponível o cliente irá ser processado por tempo definido pela função normal de média 4.5 minutos e desvio 2 minutos, e sair do simulador. O simulador irá ser executado por 1560 minutos.

A tabela abaixo demonstra os resultados obtidos pelo simulador GPSS e pelo modelo Java para tempo de simulação de 26 horas:

	Tempo médio espera	Comp. Med, fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos
Java	7,283	5,952	23	94,175%	1276
GPSS	4.159	3,410	19	91,7%	1279

Mais uma vez o GPSS apresenta valores semelhantes ao simulador JAVA. A taxa de utilização de serviço, o número de clientes atendidos e o comprimento máximo e médio da fila são bastante semelhantes em ambos modelos. Podemos portanto concluir que, pela comparação dos resultados, o simulador criado é válido.

Conclusões da Validação:

Sendo que o código produzido do simulador em Java teve resultados satisfatórios em todas as validações e critérios usados, concluímos que produz resultados fidedignos, representando de forma adequada o sistema real que pretendemos estudar e portanto é um simulador válido.

4. Análise Comportamento dos sistemas

4.1 Resultados do sistema atual, *cenário base*, para 26 horas:

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	248,8083	165,288	323	99,881%	713	324
Gasóleo	4,053	0,621	5	62,559%	239	0
Loja	0,559	0,341	3	64,213%	951	0

4.2 Resultados *cenário i,* para 26 horas:

	Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
Gasolina	12,084	8,024	25	96,212%	1034	2
Gasóleo	4,071	0,621	5	62,541%	238	0
Loja	1,783	1,452	7	85,687%	1271	0

4.3 Resultados c*enário ii,* para 26 horas:

Tempo médio espera	Comp.medio fila	Comp. máximo fila	Utilização Serviço	Nº Clientes atendidos	Nº Clientes na fila
7,283	5,952	23	94,175%	1276	0

4.4 Análise conclusões

Analiticamente, de acordo com os valores propostos no enunciado para o *cenário base*, seria de esperar que, com apenas dois postos na gasolina, os recursos existentes não fossem suficientes para atender à grande afluência de clientes. Sabemos que a taxa de utilização é definida por:

$$U = \frac{número\ de\ clientes * tempo\ de\ serviço}{tempo\ de\ simulação * número\ de\ atendedores}$$

Assim, se simularmos para 1 hora, considerando que chegam aproximadamente 50 clientes nesse tempo de simulação, obtemos:

$$U_{gasóleo} = \frac{10*4}{60*1} < 1 \qquad U_{gasolina} = \frac{40*4}{60*2} > 1 \qquad U_{loja} = \frac{50*1}{60*1} < 1$$

Vemos que, com apenas dois postos atribuídos à gasolina haverá um problema de utilização. Isto é reforçado pelos resultados obtidos que apresentamos na tabela, onde observamos que a taxa de utilização da seção de gasolina está muito próxima de 100%, enquanto que as restantes áreas da estação serviço apresentam taxas de 63% e 64% aproximadamente. Este fator provoca filas exageradamente grandes e consequentemente tempos de espera muito elevados para o serviço de gasolina, o que implica que muitos clientes não chegam a ser atendidos até ao tempo final da simulação, o que não seria viável numa situação real.

Os restantes cenários propostos irão resolver o problema do *cenário base.* No caso do *cenário i*, o ajuste efetuado consiste no incremento do número de postos no serviço de gasolina. Esta alteração, como podemos ver nas tabelas acima, irá diminuir a taxa de utilização da gasolina, para o mesmo tempo de simulação, tornando a estação de serviço mais eficiente. Embora a utilização da gasolina seja ainda elevada, cerca de 96%, os tempos médios de espera na fila da gasolina são bastante inferiores aos anteriores, rondando os 12 minutos, o que permite atender mais clientes, mais 300 que no cenário base. Ainda, neste cenário, vemos que os valores do serviço de gasóleo se mantêm essencialmente inalterados, e que a utilização da loja aumenta, devido ao maior número de clientes processados, sem que os valores ultrapassem o aceitável em situação real: tempo médio de espera de cerca de 2 minuto.

No *cenário ii,* a modificação proposta consiste em alterar o esquema da estação de serviço e transformá-la num sistema completamente self-service com 4 postos disponíveis. Nesta versão, a taxa de utilização do serviço ronda os 94%. Podemos verificar também, que o comprimento máximo da fila é 23, valor ligeiramente inferior ao do *cenário i* e que o tempo médio de espera será cerca de 7 minutos.

Comparando todos os resultados, o *cenário ii* deverá ser o melhor. Tendo em conta que, a maioria dos clientes (80%) utiliza o serviço de gasolina, vemos que a sua taxa de utilização no *cenário base* e *cenário i* é superior à taxa do serviço do *cenário ii*, podemos verificar ainda, que a média do tempo de espera total no *cenário i* será:

$$12{,}084 * 0{,}8 + 4{,}071 * 0{,}2 + 1{,}783 \approx 12{,}26\ minutos$$

Este valor é superior ao do *cenário ii*, é de notar também, que no cenário ii não existem, clientes na fila, isto é, todos os clientes que chegaram foram atendidos durante o tempo de simulação, comprovando que o *cenário ii* será mais eficiente.

5. Rentabilidade das soluções

Para determinar o lucro de cada cenário de acordo com os meses, tirámos partido dos métodos criados para o cálculo do balanço económico da estação de serviço. Calculando a cada mês os custos salariais, receitas e balanço de cada cenário. Para facilitar estas operações considerámos que os meses têm sempre 30 dias.

O primeiro cenário, *cenário i*, representa um investimento de 30000€ sendo que o valor total dos salários mensais dos 15 funcionários (5 postos existentes e 3 turnos) amonta a 7500€.

No *cenário ii*, o investimento será de 105000€, porém o montante total mensal gasto em salários é 3000€, pois são pagos apenas 6 salários (2 funcionários a supervisionar por turno, havendo 3 turnos). Desta forma, a longo prazo espera-se que o *cenário ii* seja mais rentável.

Para determinar quando o *cenário ii* ultrapassará o *cenário i* em termos de rentabilidade deve-se passar como *input* na interface valores de tempo de simulação em horas que correspondam ao número de meses desejados, já que o código do cálculo foi idealizado para tal.

	Cenário i			Cenário ii		
Mês	**Custos salariais**	**Receitas**	**Balanço**	**Custos salariais**	**Receitas**	**Balanço**
1	7500,0 €	54484,5 €	16984,5 €	3000,0 €	54484,5 €	-53515,5 €
2	15000,0 €	108186,0 €	63186,0 €	6000,0 €	108198,0 €	-2802,0 €
3	22500,0 €	162309,0 €	109809,0 €	9000,0 €	162318,0 €	48318,0 €
4	30000,0 €	216474,0 €	156474,0 €	12000,0 €	216456,0 €	99456,0 €
5	37500,0 €	270523,5 €	203023,5 €	15000,0 €	269886,0 €	149886,0 €
6	45000,0 €	324067,5€	249067,5 €	18000,0 €	324046,5 €	201046,5 €
7	52500,0 €	377772,0 €	295272.0€	21000,0 €	377779,5 €	251779,5 €
8	60000,0 €	432016,5 €	342016,5 €	24000,0 €	432022,5 €	303022,5 €
9	67500,0 €	485568,5 €	388068,5 €	27000,0 €	485563,5 €	353563,5 €
10	75000,0 €	539200,5€	434200,5 €	30000,0 €	539194,5 €	404194,5 €
11	82500,0 €	593317,5 €	480817,0 €	33000,0 €	593337,0 €	455337,0 €
12	90000,0 €	647656,5 €	527656.5 €	36000,0 €	647665,5 €	506665,5 €
13	97500,0 €	701874,0 €	574374,0 €	39000,0 €	701877,0 €	557877,0 €
14	105000,0 €	755814,0 €	620814,0 €	42000,0 €	755797,5 €	608797,5 €
15	112500,0 €	809290,5 €	666790,5 €	45000,0 €	809290,5 €	659290,5 €
16	120000,0 €	863293,5 €	713293,5 €	48000,0 €	863302,5 €	710302,5 €
17	127500,0 €	917146,5 €	**759646,5 €**	51000,0 €	917148,0 €	**761148,0 €**
18	135000,0 €	971713,5€	806713,5€	54000,0 €	971718,0 €	812718,0 €

Como podemos verificar aos 17 meses o *cenário ii* torna-se mais rentável que o *cenário i*, ou seja, o self-service demora 1 ano e 5 meses a ser mais lucrativo do que a instalação de mais um posto de gasolina. Ainda, ambas as soluções adotadas depressa recuperam o valor do seu investimento inicial e passam a ser lucrativos, no caso do *cenário ii* isto acontece depois dos 2 meses e para o *cenário i* antes do primeiro mês.

6. Conclusões Gerais

1) As simulações de cenários permitem auxiliar em tomadas de decisões importantes, já que possibilitam uma avaliação fidedigna dos benefícios e possíveis otimizações de determinadas situações antes de modificar o sistema real.

2) Todas as simulações devem ser validadas de forma a garantir que são confiáveis e os seus resultados estão de acordo com o esperado. É importante produzir simuladores fidedignos

já que muitas decisões são tomadas com base neles, porque é comum as situações não serem testadas na realidade antes de serem implementadas.

3) Devem-se usar parâmetros o mais próximos possíveis da realidade, de modo a assegurar a validade e correção das simulações efetuadas.

7. Distribuição de tarefas

A maioria do trabalho desenvolvido foi efetuada nas aulas, em grupo por ambos elementos. Aqui apresentamos uma tabela simplificada das tarefas efetuadas e das suas distribuições. As tarefas que contêm os nomes das duas alunas foram executadas em tempo de aula, com a supervisão da docente ou discutidas em grupo. No caso do relatório, a pessoa responsável pela tarefa em questão desenvolveu essa parte do relatório e as restantes foram também efetuadas e discutidas em grupo.

	Código modificado	**Interface**	**Validação interna cenário i**	**Validação interna cenário ii**	**Resultados cenário i**	**Resultados cenário ii e alínea d)**
Alexandra						
Rita						

	Código GPSS	**Formular hipóteses**	**Análise de resultados e comportamento**
Alexandra			
Rita			

Código em Anexo:

Simulador.java

```
public class Simulador extends JFrame{

   public JPanel jp = new JPanel() ;

   String resultados;
   private double instante;
   private double tempoS;
   private double media_cheg;
   private int n_clientes;
   private Servico loja;
   private Servico gasoleo;
   private Servico gasolina;
   private Servico selfService;
   private ListaEventos lista;
   private int cenario;

   public Simulador(int cenario, int tempo, double media_cheg, double m_servLoja, double desvioLoja,double m_servGasolina, double desvioGasolina,double m_servGasoleo, double desvioGasoleo, int funcL, int funcGasoleo, int funcGasolina ) {
      this.cenario=cenario;
      this.media_cheg = media_cheg;
      this.tempoS=tempo;
      instante = 0;
      lista = new ListaEventos(this);
      if(cenario==1){
         loja = new Servico (this,funcL,"LOJA", m_servLoja, desvioLoja, 50);
         gasolina = new Servico (this,funcGasolina,"GASOLINA", m_servGasolina, desvioGasolina, 10);
         gasoleo = new Servico (this,funcGasoleo,"GASOLEO", m_servGasoleo, desvioGasoleo, 30);
         insereEvento (new Chegada(instante, this, gasolina));
      }
      else{
         selfService = new Servico (this,funcL,"SELFSERVICE", m_servLoja, desvioLoja, 30);
         insereEvento (new Chegada(instante, this, selfService));
      }
   }

   public static void main(String[] args) {
      Interf inte = new Interf();
   }

   void insereEvento (Evento e1){
      lista.insereEvento (e1);
   }

   private void act_stats(){
      if(cenario==1){
         loja.act_stats();
         gasolina.act_stats();
         gasoleo.act_stats();
      }
      else{
         selfService.act_stats();
      }
   }

   private void relat (){
      resultados="";
      if(cenario==1){
```

```
        resultados= resultados+""+gasolina.relat();
        resultados= resultados+""+gasoleo.relat();
        resultados= resultados+""+loja.relat();
        gasolina.act_stats();
        gasoleo.act_stats();
    }
    else{
        resultados= resultados+""+selfService.relat();
        selfService.act_stats();
    }
}

public void executa (Interf inte){
    Evento e1;
    while (instante< tempoS){
        e1 = (Evento)(lista.removeFirst());
        instante = e1.getInstante();
        act_stats();
        e1.executa();
    };
    relat();
    inte.resultados=resultados;
}

public double getInstante() {
    return instante;
}
public double getMedia_cheg() {
    return media_cheg;
}
public double getReceitas(){
    double money=0;
    if(cenario==1){
        money=loja.getAtendidos()*1.5;
    }
    else{
        money = selfService.getAtendidos()*1.5;
    }
    return money;
}
public double getDespesas(){
    double salarios=0;
    double mes =tempoS/60/24/30;
    int funcs=0;

    if(cenario==1){
        funcs= 3*(loja.getFuncionarios()+ gasolina.getFuncionarios()+ gasoleo.getFuncionarios());
    }
    else{
        funcs=3* 2;
    }
    salarios = 500*mes*funcs;
    return salarios;
}
public double getLucro(){
    double saldo=0;
    double investimento=30000;
    if(cenario!=1){
        investimento*=3.5;
    }
    saldo=getReceitas()-(getDespesas()+investimento);
    return saldo;
```

```
    }
    public Servico getLoja() {
        return loja;
    }
    public Servico getGasoleo() {
        return gasoleo;
    }
    public Servico getSelfService() {
        return selfService;
    }
    public Servico getGasolina() {
        return gasolina;
    }
    public int getCenario() {
        return cenario;
    }
    public void limpa(){
        if(cenario==1){
            gasolina.clearTudo();
            gasoleo.clearTudo();
            loja.clearTudo();
        }
        else{
            selfService.clearTudo();
        }
    }

}
```

Servico.java

```
public class Servico {
    private String nome;
    private int estado;
    private int atendidos;
    private double temp_ult, soma_temp_esp, soma_temp_serv;
    private int maxFila;
    private Vector<Cliente> fila;
    private Simulador s;
    private int funcionarios;
    private double med_serv, desvio;
    private int seed;
    private double [] v = new double [2];
    private boolean calc = true;

    Servico (Simulador s, int funcionarios, String nome, double med_serv, double desvio, int seed){
        this.s = s;
        this.nome = nome;
        fila = new Vector<Cliente>();
        estado = 0;
        temp_ult = s.getInstante();
        atendidos = 0;
        soma_temp_esp = 0;
        soma_temp_serv = 0;
        this.funcionarios = funcionarios;
        this.seed= seed;
        this.med_serv=med_serv;
        this.desvio=desvio;
        this.maxFila=0;
    }

    public void insereServico (Cliente c){
```

```
    if (estado < funcionarios) {
        estado ++;
        if(s.getCenario()==1){
            if("LOJA".equals(nome)){
                if (calc){
                    v = Aleatorio.normal(med_serv,desvio,seed);
                    calc = false;
                   }
                else {
                    calc = true;
                    v[0] = v[1];
                }
                s.insereEvento (new Saida(s.getInstante()+v[0], s, this));
            }
            else{
                if (calc){
                    v = Aleatorio.normal(med_serv,desvio,seed);
                    calc = false;
                   }
                else {
                    calc = true;
                    v[0] = v[1];
                }
                s.insereEvento (new Transfere(s.getInstante()+ v[0], s, this, c));
            }
        }
        else{
            if (calc){
                if(c.getTipo()==1){
                    v = Aleatorio.normal(med_serv,desvio,10);
                }
                else{
                    v = Aleatorio.normal(med_serv,desvio,20);
                }
                calc = false;
               }
            else {
                calc = true;
                v[0] = v[1];
            }
            s.insereEvento (new Saida(s.getInstante()+v[0], s, this));
        }
    }
    else{
        if(fila.size()>maxFila){
            maxFila=fila.size();
        }
        fila.addElement(c);
    }

}

public Cliente removeServico (){
    atendidos++;
    if (fila.isEmpty()) estado --;
    else {
        Cliente c = (Cliente)fila.firstElement();
        fila.removeElementAt(0);
        if(s.getCenario()==1){
            if("LOJA".equals(nome)){
                if (calc){
                    v = Aleatorio.normal(med_serv,desvio,seed);
```

```
                calc = false;
              }
              else {
                calc = true;
                v[0] = v[1];
              }
             s.insereEvento (new Saida(s.getInstante()+v[0], s, this));
          }
          else{
              if (calc){
                v = Aleatorio.normal(med_serv,desvio,seed);
                calc = false;
              }
              else {
                calc = true;
                v[0] = v[1];
              }
             s.insereEvento (new Transfere(s.getInstante()+ v[0], s, this, c));
          }
        }
        else{
           if (calc){
              if(c.getTipo()==1){
                 v = Aleatorio.normal(med_serv,desvio,10);
              }
              else{
                 v = Aleatorio.normal(med_serv,desvio,20);
              }
              calc = false;
             }
           else {
              calc = true;
              v[0] = v[1];
           }
           s.insereEvento (new Saida(s.getInstante()+v[0], s, this));
        }
        return c;
      }
      return null;
   }

   public void act_stats(){
      double temp_desde_ult = s.getInstante() - temp_ult;
      temp_ult = s.getInstante();
      soma_temp_esp += fila.size() * temp_desde_ult;
      soma_temp_serv += estado * temp_desde_ult;
   }

   public String relat (){
      double temp_med_fila = soma_temp_esp / (atendidos+fila.size());
      double comp_med_fila = soma_temp_esp / s.getInstante();
      double utilizacao_serv = soma_temp_serv / s.getInstante();
      String resul="";
      resul=resul+nome+"\n"+"Tempo medio espera: "+String.format("%.3f" , temp_med_fila)+"\n Comp. medio fila: "+String.format("%.3f" , comp_med_fila)+"\n Comp. maximo fila: "+maxFila+"\n Utilização Serv: "+String.format("%.3f" , (utilizacao_serv/funcionarios)*100)+"%"+"\n Numero de Clientes atendidos: "+atendidos+"\n Numero de Clientes na fila:"+fila.size()+"\nTempo de Simulação:"+String.format("%.3f" , s.getInstante())+"\n\n";
      return resul;
   }

   public int getAtendidos() {
      return atendidos;
```

```
    }
    public String getNome() {
        return nome;
    }
    public int getFuncionarios(){
        return funcionarios;
    }
    public Vector<Cliente> getFila() {
        return fila;
    }
    public void clearTudo(){
        fila.clear();
        atendidos=0;
        temp_ult=0.0;
        soma_temp_esp=0;
        soma_temp_serv=0;
        maxFila=0;
    }
}
```

Evento.java

```
public abstract class Evento {

    protected double instante;
    protected Simulador s;
    protected Servico servico;

    Evento (double i, Simulador s, Servico servico){
        instante = i;
        this.s = s;
        this.servico = servico;
    }
    public boolean menor (Evento e1){
        return (instante < e1.instante);
    }

    abstract void executa ();

    public double getInstante() {       return instante;    }
    public Servico getServico() {       return servico;    }

}
```

Chegada.java

```
public class Chegada extends Evento {

    Chegada (double i, Simulador s, Servico servico){
        super (i, s, servico);
    }

    void executa (){
        Cliente cl= new Cliente();
        if(s.getCenario()==1){
            if(cl.getTipo()==1){
                s.getGasolina().insereServico(cl);
            }
            else{
                s.getGasoleo().insereServico(cl);
            }
        }
        else{
            s.getSelfService().insereServico(cl);
        }
```

```
        s.insereEvento (new Chegada(s.getInstante()+Aleatorio.exponencial(s.getMedia_cheg()), s, servico));
    }

    public String toString(){
        return "Chegada em " + instante;
    }

}
```

Saida.java

```
public class Saida extends Evento {

    private Servico servico;

            Saida (double i, Simulador s, Servico servico){
                    super(i, s, servico);
                    this.servico=servico;
    }
    void executa (){
        servico.removeServico();
    }

    public String toString(){
        return "Saída em " + instante;
    }

}
```

Cliente.java

```
public class Cliente {

    private int tipo;

    Cliente(){
        tipo=setTipo();
    }
    private int setTipo(){
        double var=0;
        var=RandomGenerator.rand(0);
        if(var<0.2){
            return 2;
        }
        else{
            return 1;
        }
    }
    public int getTipo(){       return tipo;    }

}
```

Transfere.java

```
public class Transfere extends Evento{

    private Servico servico;
    private Cliente cl;

    public Transfere(double i, Simulador s, Servico servico, Cliente cl) {
        super(i, s, servico);
        this.cl = cl;
        this.servico=servico;
    }
```

```
    void executa() {
        Cliente cl = servico.removeServico();
        s.getLoja().insereServico(cl);
    }

}
```

Aleatorio.java

```
public class Aleatorio {

    static public int flag=0;
    static public double v1, v2, w, valor;

    static double exponencial (double m){
        return (-m*Math.log(RandomGenerator.rand64(40)));
    }
    static double [] normal (double m,double d, int seed){
        double [] y = new double [2];
                    y[0]=-1;
                    y[1]=-1;
        while(y[0]<0 || y[1]<0){
            do {
                v1=2*RandomGenerator.rand64(seed)-1;
                v2= 2*RandomGenerator.rand64(seed)-1;
                w= Math.pow(v1, 2)+Math.pow(v2, 2);
            } while(w>1);
                  y[0]=v1 * Math.sqrt(-2*Math.log(w)/w);
                  y[1]= v2 * Math.sqrt(-2*Math.log(w)/w);
                  y[0]=m+y[0]*d;
                  y[1]=m+y[1]*d;
        }
        return y;
    }

}
```