Introdução à Simulação Discreta

Mauricio Pereira dos Santos
Departamento de Matemática Aplicada
Instituto de Matemática e Estatística

UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

Editoração: O autor, criando arquivo texto no format LaTex.
Gráfico da Randu gerado no Matlab e inserido no texto como EPS (Encapsulated Postcript File).
Telas do Arena inseridas no texto como EPS (Encapsulated Postcript File).
Compiladores usados nos programas: Turbo Pascal e Visual Basic 6.

070316

Prefácio

O objetivo deste trabalho é fornecer aos alunos das cadeiras de Simulação da UERJ um referencial que os auxilie no estudo e acompanhamento da matéria.
A literatura, sobre o assunto, na língua portuguesa é bastante reduzida e, como agravante, algumas das poucas obras existentes estão esgotadas.
Mesmo a literatura em inglês sobre Simulação é pequena, se comparada com outros tópicos que formam a chamada Pesquisa Operacional.
Por sua vez os livros genéricos de Pesquisa Operacional, quase sem exceção, apresentam apenas um capítulo sobre Simulação sendo o assunto, como não poderia deixar de ser, tratado de forma bastante superficial.
O estudo dos modelos de Simulação implica, obrigatoriamente, no uso do computador. A falta dele limita, severamente, a compreensão das técnicas utilizadas. Por esta razão incluímos no texto programas que possam ajudar aos alunos a fazer suas próprias aplicações.
Atualmente, na UERJ, a cadeira de Processamento de Dados ensina como linguagem de programação, o Pascal, razão pela qual os programas do texto são apresentados naquela linguagem. Para confecção dos programas, o compilador usado foi o Turbo Pascal mas qualquer outro compilador Pascal deve poder ser usado sem maiores complicações. No capítulo 5 é apresentado um programa que foi feito para o ambiente windows, utilizando-se o Visual Basic. Este programa é um dos módulos do programa PO que contém módulos de várias aplicações de Pesquisa Operacional.

Agradecemos a todos aqueles que nos ajudaram nesta tarefa, especialmente os alunos que foram cobaias no uso dos rascunhos deste trabalho.
De antemão agradeço desde já a todos aqueles que puderem apontar imperfeições e erros que possam ser corrigidos.

O Autor

Conteúdo

Capítulo 1

Introdução à Simulação

Uma simulação é a imitação, durante determinado período de tempo, da operação de um sistema ou de um processo do mundo real. Feita a mão (estudando) ou em um computador, a simulação envolve a geração de uma história artificial do sistema, e a partir desta história artificial a inferência de como o sistema real funcionaria. O comportamento do sistema é estudado pela construção de um Modelo de Simulação. Este modelo normalmente toma a forma de um conjunto de considerações relacionadas a operação do sistema. Estas considerações são expressas através de relações matemáticas, lógicas e simbólicas entre as entidades, ou objetos de interesse, do sistema. Uma vez construído e validado, um modelo pode ser usado para investigar uma grande quantidade de questões do tipo "e se..." sobre o sistema do mundo real. Alterações no sistema podem ser inicialmente simuladas para se prever as conseqüências no mundo real. A Simulação também pode ser usada para estudar sistemas no estágio de projeto, ou seja antes do sistema ser construído. Assim, a Simulação pode usada tanto como uma ferramenta de análise para prever o efeito de mudanças em sistemas já existentes, quanto como uma ferramenta para prever a performance de novos sistemas sobre as mais variadas circunstâncias.

1.1 Vantagens e Desvantagens da Simulação

As vantagens principais da simulação são:

- Novas políticas, procedimentos operacionais, regras de negócio, fluxos de informação, etc..., podem ser estudadas sem se alterar o mundo real.
- Novos equipamentos, layouts, sistemas de transporte, etc..., podem ser testados sem se comprometer recursos na sua aquisição.
- Hipóteses sobre como e porque certos fenômenos ocorrem podem ser testados visando verificar sua praticabilidade.
- O tempo pode ser comprimido ou expandido permitindo acelerar ou retardar o fenômeno sob investigação.
- Pode-se entender melhor sob a interação das variáveis do sistema.

- Pode-se entender melhor a participação das variáveis na performance do sistema.
- Um modelo de simulação pode ajudar a entender como um sistema funciona como um todo, em relação a como se pensa que o sistema opera individualmente.
- Questões do tipo "e se..." podem ser respondidas. Isto é extremamente útil na fase de design de um projeto.

As desvantagens a serem consideradas são:

- A construção de Modelos de Simulação requer treinamento especial. É uma arte que é aprendida com tempo e experiência. Além disto se 2 modelos são construídos por 2 profissionais competentes, eles terão semelhanças, mas será altamente improvável que sejam iguais.
- Os resultados de uma Simulação podem ser difíceis de interpretar. Como a maioria das saídas de uma simulação são variáveis aleatórias (elas estão normalmente baseadas em entradas aleatórias), é difícil determinar se uma observação é o resultado do relacionamento entre as variáveis do sistema ou conseqüência da própria aleatoriedade.
- A construção e análise de Modelos de Simulação pode consumir muito tempo e, como conseqüência, muito dinheiro. Economizar por sua vez pode levar a modelos incompletos.
- A Simulação é usada em muitos casos onde uma solução analítica é possível. A simulação não dá resultados exatos.

1.2 Áreas de aplicação

Existem inúmeras áreas de aplicação da simulação. A seguir estão listadas algumas das mais importantes:

- Simulação das operações de uma companhia aérea para testar alterações em seus procedimentos operacionais.
- Simulação da passagem do tráfego em um cruzamento muito grande, onde novos sinais estão para ser instalados.
- Simulação de operações de manutenção para determinar o tamanho ótimo de equipes de reparo.
- Simulação de uma siderúrgica para avaliar alterações nos seus procedimentos operacionais.
- Simulação da economia de um setor de um país para prever o efeito de mudanças econômicas.

- Simulação de batalhas militares visando avaliar o desempenho de armas estratégicas.
- Simulação de sistemas de distribuição e controle de estoque, para melhorar o funcionamento destes sistemas.
- Simulação de uma empresa como um todo para avaliar o impacto de grandes mudanças ou como treinamento para seus executivos (Business Games).
- Simulação de sistemas de comunicações para determinar o que é necessário para fornecer um determinado nível de serviço.
- Simulação de uma barragem em um determinado rio para avaliar os problemas advindos com a sua construção.
- Simulação de uma linha de produção em determinada indústria, para avaliar efeitos de mudanças previstas no processo produtivo ou diminuição de gargalos na linha de produção.

1.3 Componentes de um Sistema

Um **Sistema** é definido como um grupo de objetos que estão juntos em alguma interação ou interdependência, objetivando a realização de algum objetivo. Um exemplo poderia ser um sistema de produção de automóveis. As máquinas, componentes, peças e trabalhadores operam em conjunto, em uma linha de montagem, visando a produção de veículos.
De forma a entender e analisar um sistema, alguns termos precisam ser definidos:
Uma **Entidade** é um objeto de interesse no sistema.
Um **Atributo** é uma propriedade de uma entidade.
Uma **Atividade** representa um período de tempo de certo tamanho.
O **Estado** do sistema é definido como sendo a coleção de variáveis necessárias para descrever o sistema em um dado instante.
Um **Evento** é definido como uma ocorrência instantânea que pode mudar o estado do sistema.
O termo **Endógeno** é usado para descrever atividades e eventos ocorrendo dentro do sistema e **Exógeno** é usado para descrever atividades e eventos que ocorrem fora do sistema.
A tabela a seguir mostra alguns exemplos para os termos definidos acima:

Sistema	Exemplo Entidade	Exemplo Atributo	Exemplo Atividade	Exemplo Evento	Exemplo Variáveis Estado
Banco	Clientes	Saldo na C/C	Depositar	Chegada a Agência	Nº Clientes Esperando
Produção	Máquinas	Taxa Quebra	Soldagem	Quebra	Máquinas Paradas
Comunicação	Mensagens	Tamanho	Transmissão	Chegada	Mensagens Esperando
UERJ	Aluno	CR	Matrícula	Cadeira Cancelada	Nº Alunos Matriculados

1.4 Tipos de Modelos

Modelos de Simulação podem ser **Estáticos** ou **Dinâmicos**.
Um modelo de simulação estática, algumas vezes chamado de Simulação de Monte Carlo, é um modelo onde a passagem do tempo é irrelevante.
Modelos de Simulação Dinâmicos representam sistemas cujos resultados variam com a passagem do tempo.
Um modelo de simulação pode ser ainda **Determinístico** ou **Estocástico**.
Modelos de simulação que não contém nenhuma variável aleatória são classificados como determinísticos, ou seja, para um conjunto conhecido de dados de entrada teremos um único conjunto de resultados de saída.
Um modelo estocástico de simulação tem uma ou mais variáveis aleatórias como entrada. Estas entradas aleatórias levam a saídas aleatórias que podem somente ser consideradas como estimativas das características verdadeiras de um modelo.
Assim, por exemplo, a simulação (estocástica) do funcionamento de uma agência bancária envolve variáveis aleatórias como o intervalo entre chegadas e a duração dos serviços prestados. Logo, medidas como o número médio de clientes esperando e o tempo médio de espera de um cliente, devem ser tratadas como estimativas estatísticas das medidas reais do sistema.

1.5 Modelos Discretos e Contínuos

Os modelos de simulação dinâmicos podem ser **Discretos** ou **Contínuos**. Em uma simulação discreta, considera-se somente os eventos onde há alteração do sistema, ou seja, o tempo decorrido entre alterações do estado do sistema não é relevante para **a obtenção dos resultados da simulação**, embora o tempo nunca pare. Alguns autores a chamam de **Simulação de Eventos Discretos**, enfatizando assim que a discretização se refere apenas à ocorrência dos eventos ao longo do tempo.
Um exemplo seria a simulação de uma agência bancária onde entre a chegada (ou a saída) de clientes o estado do sistema não se altera.
Numa Simulação Contínua o sistema se altera a cada fração de tempo. Exemplos clássicos são a simulação de um avião voando e a passagem de água por uma barragem.

1.6 Etapas de um projeto de simulação

As etapas básicas de um projeto de simulação são:

1. Formulação do problema

   Cada projeto deve começar com a definição do problema a ser resolvido. É importante que a definição esteja clara para todos que participam do projeto.

2. Determinação dos objetivos e planejamento global do projeto

   O objetivo indica as questões que devem ser respondidas pela simulação. Neste ponto deve ser considerado se a simulação é a metodologia apropriada para o problema. Nesta fase deve-se fazer também uma estimativa do tamanho da equipe envolvida, custo, tempo, etc...

3. Construção do Modelo

   A construção de um modelo de um sistema é provavelmente mais arte que ciência. Embora não seja possível fornecer um conjunto de instruções que possibilitem construir à cada vez modelos apropriados, existem algumas linhas mestres que podem ser seguidas. A arte de modelar é melhorada se conseguimos extrair as partes essenciais de um problema, selecionar e modificar as considerações básicas que caracterizam o sistema e então enriquecer e elaborar o modelo até a aproximação de resultados úteis. Assim é melhor começar com um modelo simples e ir aumentando sua complexidade. Entretanto a complexidade do modelo não necessita exceder o necessário para acompanhar os propósitos para qual o modelo foi construído. Não é necessário se ter uma relação de um para um entre o modelo e o sistema real. Somente a essência do sistema real é necessária. É indispensável envolver o usuário na construção do modelo. Isto faz com que a qualidade do modelo resultante fique melhor e aumenta a confiança do usuário na sua futura aplicação. Somente exercícios e a prática ajudam na construção de modelos melhores.

4. Coleta de dados

   Há uma interação constante entre a construção de um modelo e a coleta dos dados de entrada necessários. Geralmente quanto mais complexo o modelo, mais dados são necessários. Como a coleta de dados toma um tempo muito grande do tempo total de um projeto de simulação, é necessário começar esta coleta o mais cedo possível.

5. Codificação

   Como a maioria dos sistemas do mundo real resulta em modelos que requerem um grande número de informações e de cálculos, o modelo deve ser programado em um computador digital. O modelador deve decidir se programa em uma linguagem de programação comum como JAVA, C, PASCAL, BASIC etc..., ou se usa um pacote como o ARENA, SIMUL, PROMODEL, CRYSTAL BALL, etc... Como codificar um modelo leva, normalmente, muito tempo mesmo se a equipe possui bons programadores, a tendência atual no mercado é se trabalhar com pacotes.
   Temos que mencionar também o uso de planilhas (EXCEL) para se construir modelos de simulação.

6. Testes

   Após a codificação dos programas é necessário testá-los para verificar se eles não tem algum erro de programação. Deve-se preparar um conjunto de dados com a finalidade exclusiva de se testar os programas.

7. Validação

   Nesta fase se verifica se o modelo é uma representação precisa do sistema que se quer modelar. É nesta fase que se faz a chamada calibração do modelo, ou seja, são feitos ajustes até que os resultados nos dêem garantias de que o modelo é uma boa representação do problema sendo modelado.

8. Produção

   Nesta etapa o modelo é colocado em produção e os dados obtidos são analisados. A produção pode envolver a execução, várias vezes, do modelo, variando-se os dados e os parâmetros de entrada.

9. Avaliação global dos resultados

   Nesta fase avalia-se se os resultados obtidos estão condizentes com os esperados. Caso sejam encontradas discrepâncias podemos ter que voltar à etapa de construção do modelo.

10. Documentação e implementação É fundamental, como em qualquer projeto, que a simulação seja documentada de forma clara e concisa. Os resultados obtidos também devem ser documentados e arquivados. A implantação, se o usuário participou do processo, tende a ser bem mais simples do que nos casos em que o usuário não teve uma participação ativa.

1.7 Exemplos de modelos de Simulação

Para entender como funciona uma simulação, vamos ver alguns exemplos simples:

1.7.1 Quebra de rolamentos

Uma grande máquina industrial tem 3 rolamentos diferentes que quebram de tempos em tempos. A probabilidade da vida útil (em horas de operação) de um rolamento está dada na tabela abaixo:

Vida do Rolamento (horas)	Probabilidade
1.000	0.10
1.100	0.13
1.200	0.25
1.300	0.13
1.400	0.09
1.500	0.12
1.600	0.02
1.700	0.06
1.800	0.05
1.900	0.05

Quando um rolamento quebra, a máquina para e um mecânico é chamado para instalar um novo rolamento no lugar do que quebrou.

O tempo que o mecânico demora para chegar ao rolamento quebrado também é uma variável aleatória, com a distribuição dada na tabela abaixo:

Tempo de espera (minutos)	Probabilidade
5	0.60
10	0.30
15	0.10

Cada minuto que a máquina fica parada custa $5 e o custo do mecânico é de $1/minuto trabalhado substituindo rolamento. O mecânico demora 20 minutos para trocar 1 rolamento, 30 minutos para trocar 2 e 40 minutos para trocar os 3. Cada rolamento novo custa $20. Alguém sugeriu que ao quebrar um dos rolamentos, se fizesse logo a troca dos 3. Deseja-se avaliar a situação do ponto de vista econômico.

Solução

Temos que comparar o custo da alternativa atual e da alternativa proposta. Precisamos estabelecer um horizonte de tempo para fazer esta comparação. Considerando que a menor vida útil de um rolamento é 1.000 horas (mais de 1 mês), vamos estabelecer um horizonte de 20.000 horas (um pouco mais de 2 anos) para fazer a comparação.
Como a vida útil dos rolamentos e a espera pelo mecânico são variáveis aleatórias que seguem as distribuições vistas anteriormente, temos que relacionar àquelas distribuições com uma tabela de números aleatórios.
Assim sendo, vamos imaginar que temos um gerador de números aleatórios capaz de gerar qualquer inteiro entre 0 e 99, ou seja 100 números. Vamos atribuir a cada duração de vida útil uma faixa destes números que me garanta que a distribuição probabilística seja mantida.
Como a 1ª vida útil (1.000 horas) tem 10% de probabilidade de ocorrer, vamos atribuir a esta duração a faixa de 0 a 9 inclusive, ou seja 10 números (10% dos 100 números). Para a 2ª duração provável (1.100 horas), com 13% de probabilidade de ocorrência, vamos atribuir a faixa de 10 a 22 inclusive, ou seja 13 números. Podemos continuar para as demais durações prováveis dos rolamentos como pode ser visto na tabela a seguir, ressaltando que a probabilidade acumulada dá o limite das faixas escolhidas.

Vida do Rolamento (horas)	Probabilidade	Probabilidade Acumulada	Nº Aleatório Atribuído
1.000	0.10	0.10	0 – 9
1.100	0.13	0.23	10 – 22
1.200	0.25	0.48	23 – 47
1.300	0.13	0.61	48 – 60
1.400	0.09	0.70	61 – 69
1.500	0.12	0.82	70 – 81
1.600	0.02	0.84	82 – 83
1.700	0.06	0.90	84 – 89
1.800	0.05	0.95	90 – 94
1.900	0.05	1.00	95 – 99

Tabela semelhante pode ser construída para a espera pela chegada do mecânico.

Tempo de Espera (minutos)	Probabilidade	Probabilidade Acumulada	Nº Aleatório Atribuído
5	0.60	0.60	00 – 59
10	0.30	0.90	60 – 89
15	0.10	1.00	90 – 99

Com os dados das tabelas acima, podemos executar a simulação que, neste caso, foi realizada numa planilha **EXCEL**, apresentando os seguintes resultados para o rolamento 1:

ROLAMENTO 1					
Seqüencia	Nº Aleatório	Vida (Horas)	Vida Acumulada (Horas)	Nº Aleatório	Espera (Min)
1	62	1.400	1.400	61	10
2	85	1.700	3.100	10	5
3	89	1.700	4.800	46	5
4	24	1.200	6.000	28	5
5	99	1.900	7.900	55	5
6	27	1.200	9.100	64	10
7	89	1.700	10.800	63	10
8	12	1.100	11.900	75	10
9	2	1.000	12.900	54	5
10	34	1.200	14.100	67	10
11	7	1.000	15.100	90	15
12	75	1.500	16.600	14	5
13	22	1.100	17.700	80	10
14	97	1.900	19.600	84	10
15	37	1.200	**20.800**	9	5
					$\sum$ **=120**

Podemos observar na planilha que para cada seqüencia ou seja, rolamento novo, é gerado um número aleatório que indica qual a vida útil daquela rolamento. Tendo quebrado, após esta vida útil, o mecânico é chamado e um 2º número aleatório é gerado para definir o tempo de espera até a troca do rolamento ser iniciada.
Quando a vida acumulada ultrapassa 20.000 horas, ou seja a duração da simulação, paramos a execução do processo.
Processos semelhantes foram executados para os outros 2 rolamentos, como visto a seguir.

ROLAMENTO 2					
Seqüencia	Nº Aleatório	Vida (Horas)	Vida Acumulada (Horas)	Nº Aleatório	Espera (Min)
1	89	1.700	1.700	58	5
2	47	1.200	2.900	88	10
3	60	1.300	4.200	20	5
4	3	1.000	5.200	98	15
5	40	1.200	6.400	26	5
6	64	1.400	7.800	97	15
7	9	1.000	8.800	41	5
8	30	1.200	10.000	79	10
9	32	1.200	11.200	0	5
10	8	1.000	12.200	3	5
11	94	1.800	14.000	58	5
12	66	1.400	15.400	84	10
13	53	1.300	16.700	61	10
14	17	1.100	17.800	43	5
15	72	1.500	19.300	15	5
16	0	1.000	**20.300**	97	15
					$\sum$ **= 130**

ROLAMENTO 3					
Seqüencia	Nº Aleatório	Vida (Horas)	Vida Acumulada (Horas)	Nº Aleatório	Espera (Min)
1	49	1.300	1.300	44	5
2	26	1.200	2.500	45	5
3	2	1.000	3.500	72	10
4	83	1.600	5.100	87	10
5	21	1.100	6.200	19	5
6	20	1.100	7.300	81	10
7	60	1.300	8.600	56	5
8	34	1.200	9.800	74	10
9	63	1.400	11.200	93	15
10	69	1.400	12.600	36	5
11	44	1.200	13.800	71	10
12	76	1.500	15.300	97	15
13	55	1.300	16.600	59	5
14	85	1.700	18.300	81	10
15	21	1.100	19.400	21	5
16	5	1.000	**20.400**	1	5
					$\sum$ **=130**

Com os dados obtidos na simulação, podemos calcular o custo da situação atual:
Custo dos rolamentos = (15 + 16 + 16) × $20 = $940
Custo da máquina parada esperando pelo mecânico = (120 + 130 + 130) × $5 = $1.900
Custo da máquina parada trocando rolamento = (15 + 16 + 16) × 20 × $5 = $4.700
Custo do mecânico = (15 + 16 + 16) × 20 × $1 = $940

Custo Total = 940 + 1.900 + 4.700 + 940 = $8.480

A simulação da situação proposta apresentou os seguintes resultados:

	ROL. 1		ROL. 2		ROL. 3					
Seq.	NA	Vida hr	NA	Vida (hr)	NA	Vida (hr)	1ª Quebra	Vida Acum.	ESPERA NA	ESPERA (Min)
1	96	1.900	2	1.000	34	1.200	1.000	1.000	21	5
2	70	1.500	7	1.000	47	1.200	1.000	2.000	36	5
3	96	1.900	46	1.200	49	1.300	1.200	3.200	21	5
4	48	1.300	17	1.100	42	1.200	1.100	4.300	7	5
5	32	1.200	93	1.800	20	1.100	1.100	5.400	58	5
6	36	1.200	94	1.800	98	1.900	1.200	6.600	83	10
7	41	1.200	17	1.100	53	1.300	1.100	7.700	14	5
8	71	1.500	2	1.000	20	1.100	1.000	8.700	75	10
9	4	1.000	22	1.100	86	1.700	1.000	9.700	5	5
10	69	1.400	21	1.100	0	1.000	1.000	10.700	65	10
11	13	1.100	89	1.700	58	1.300	1.100	11.800	15	5
12	36	1.200	12	1.100	66	1.400	1.100	12.900	12	5
13	75	1.500	57	1.300	29	1.200	1.200	14.100	32	5
14	76	1.500	78	1.500	95	1.900	1.500	15.600	2	5
15	71	1.500	5	1.000	86	1.700	1.000	16.600	31	5
16	98	1.900	43	1.200	22	1.100	1.100	17.700	51	5
17	98	1.900	47	1.200	60	1.300	1.200	18.900	20	5
18	68	1.400	61	1.400	57	1.300	1.300	**20.200**	35	5
										$\sum$ =**105**

NA = Nº aleatório

Feita a simulação da situação proposta, podemos calcular os custos:
Custo dos rolamentos = (18 × 3) × $ 20 = $1.080
Custo da máquina parada esperando pelo mecânico = 105 × $5 = $525
Custo da máquina parada trocando rolamento = 18 × 40 × $5 = $3.600
Custo do mecânico = 18 × 40 × $1 = $720
Custo Total =1.080+ 525 + 3.600 + 720 = $5.925

Assim a simulação nos mostrou que a situação proposta é bem melhor em termos econômicos.

1.7.2 Fila com uma estação de serviço

Uma loja tem somente 1 atendente, o que na nomenclatura dos modelos de filas é denominado como de uma estação de serviço. Os fregueses chegam aleatoriamente com intervalo, entre eles, variando de 1 a 8 minutos. Cada valor possível do intervalo entre chegadas tem a mesma probabilidade de ocorrência, como mostrado na tabela a seguir:

Tempo entre chegadas (minutos)	Probabilidade
1	0.125
2	0.125
3	0.125
4	0.125
5	0.125
6	0.125
7	0.125
8	0.125

A duração do atendimento aos clientes varia de 1 a 6 minutos com probabilidades mostradas na tabela a seguir:

Duração do serviço (minutos)	Probabilidade
1	0.10
2	0.20
3	0.30
4	0.25
5	0.10
6	0.05

Como no exemplo anterior, temos que construir tabelas relacionando as probabilidades com números aleatórios gerados:

Tempo entre chegadas (min)	Probabilidade	Probabilidade Acumulada	Nº Aleatório Atribuído
1	0.125	0.125	000 – 124
2	0.125	0.250	125 – 249
3	0.125	0.375	250 – 374
4	0.125	0.500	375 – 499
5	0.125	0.625	500 – 624
6	0.125	0.750	625 – 749
7	0.125	0.875	750 – 874
8	0.125	1.000	875 – 999

Duração do serviço (min)	Probabilidade	Probabilidade Acumulada	Nº Aleatório Atribuído
1	0.10	0.10	0 – 9
2	0.20	0.30	10 – 29
3	0.30	0.60	30 – 59
4	0.25	0.85	60 – 84
5	0.10	0.95	85 – 94
6	0.05	1.00	95 – 99

A simulação para os primeiros 20 clientes apresentou os seguintes resultados:

	NA	Intervalo entre chegadas(min)	Instante chegada	NA	Duração serviço	Início serviço	Espera fila	Fim do serviço	Tempo total na loja(min)	Tempo ocioso atendente(min)
1			0	84	4	0	0	4	4	0
2	913	8	8	9	1	8	0	9	1	4
3	727	6	14	74	4	14	0	18	4	5
4	15	1	15	53	3	18	3	21	6	0
5	948	8	23	17	2	23	0	25	2	2
6	309	3	26	79	4	26	0	30	4	1
7	922	8	34	91	5	34	0	39	5	4
8	753	7	41	67	4	41	0	45	4	2
9	235	2	43	89	5	45	2	50	7	0
10	302	3	46	38	3	50	4	53	7	0
11	109	1	47	32	3	53	6	56	9	0
12	93	1	48	94	5	56	8	61	13	0
13	607	5	53	79	4	61	8	65	12	0
14	738	6	59	5	1	65	6	66	7	0
15	359	3	62	79	5	66	4	71	9	0
16	888	8	70	84	4	71	1	75	5	0
17	108	1	71	52	3	75	4	78	7	0
18	212	2	73	55	3	78	5	81	8	0
19	493	4	77	30	2	81	4	83	6	0
20	535	5	82	50	3	83	1	86	4	0
		$\sum =$			68		56		124	18

Podemos, a partir da simulação, inferir alguns resultados:
O tempo de espera médio de um cliente foi de 2,8 minutos. Este valor é encontrado de :

$$\begin{aligned} \text{Tempo médio de espera (min)} &= \frac{\text{Tempo total dos clientes na fila (min)}}{\text{Número total de clientes}} \\ &= \frac{56}{20} = 2,8 \text{ minutos} \end{aligned}$$

A probabilidade de que um cliente tenha que esperar na fila é 65%. Isto vem de:

$$\begin{aligned} \text{Probabilidade(espera)} &= \frac{\text{Número de clientes que esperaram}}{\text{Número total de clientes}} \\ &= \frac{13}{20} = 0,65 \end{aligned}$$

A proporção de tempo que o atendente fica ocioso é 21%. Vem de:

$$\begin{aligned}\text{Prob. do atendente estar ocioso} &= \frac{\text{Tempo total de ociosidade (min)}}{\text{Duração da Simulação}}\\ &= \frac{18}{86} = 0.21\end{aligned}$$

O atendente está ocupado $100 - 21 = 79\%$ do tempo.
O tempo de serviço médio é de 3.4 minutos. Podemos obtê-lo de:

$$\begin{aligned}\text{Tempo de serviço médio (min)} &= \frac{\text{Duração total do serviço}}{\text{Número total de clientes}}\\ &= \frac{68}{20} = 3.4 \text{ minutos}\end{aligned}$$

Este resultado pode ser comparado com o tempo de serviço esperado achando-se a média da distribuição do tempo de serviço usando a equação:

$$E(s) = \sum_{s=0}^{\infty} sp(s)$$

Temos então:
$1(0.10) + 2(0.20) + 3(0.30) + 4(0.25) + 5(0.10) + 6(0.05) = 3.2$ minutos
O resultado da simulação é um pouco maior porque o tamanho da simulação foi pequeno. Quanto maior a duração da simulação mais o resultado se aproximará de 3.2 minutos.
Alguém que fosse tomar decisões estaria interessado nos resultados obtidos acima. Obviamente seria necessário uma simulação mais demorada para se conseguir resultados mais precisos.
Entretanto, algumas inferências podem ser obtidas: A maioria dos clientes tem que esperar mas a espera não é excessiva. O atendente não fica muito tempo ocioso.
O objetivo a ser alcançado vai depender do balanço entre o custo de espera e o custo de se colocar mais atendentes.

1.7.3 O que não está explícito

Nos exemplos que vimos até aqui, alguns fatos importantes na construção de um modelo de simulação não estão devidamente explicitados.
Vimos, por exemplo, que a vida útil de um rolamento segue uma determinada distribuição. É óbvio que esta informação “não caiu do céu”. Todo um trabalho preliminar incluindo definição do tamanho da amostra, amostragem, definição da distribuição (aderência), etc, etc..., foi realizado para se chegar àquela informação.
Também ficou claro que a execução da simulação em si, por ser um processo aleatório, apresenta, normalmente, um resultado diferente a cada execução. Para que os dados obtidos pela simulação sejam estatísticamente confiáveis, também é necessário todo um trabalho para definir o número de execuções (replicações) necessárias a qualidade dos dados obtidos bem como a definição de intervalos de confiança das variáveis básicas do modelo.

Capítulo 2

A geração de Números Aleatórios

Pudemos reparar nos exemplos do capítulo anterior que a chave para simular eventos aleatórios discretos é a geração de números aleatórios. Como se usa o computador para fazer a simulação, precisamos de métodos rápidos e eficientes para gerá-los.

Os números aleatórios, gerados em computador, não são realmente aleatórios pois veremos mais adiante que eles são gerados em seqüências que podem ser reproduzidas, o que viola o princípio básico da aleatoriedade. Como contornar este fato ? Se os números passam por uma série de testes estatísticos de aleatoriedade então, para efeitos práticos, podemos considerá-los como se fossem realmente aleatórios.

Por este fato eles são conhecidos como números **Pseudo-aleatórios**. É comum se usar, em simulação, a expressão **números aleatórios** mas considere isto, <u>sempre</u>, como um sinônimo de números pseudo-aleatórios.

2.1 Propriedades desejáveis de um gerador de números aleatórios

Um gerador de números aleatórios deveria possuir todas as características abaixo:

1. Aleatoriedade

   É essencial que a seqüência gerada exiba as propriedades estatísticas dos números verdadeiramente aleatórios. Este comportamento aleatório deve ser confirmado por testes estatísticos.

2. Grande Período

   Todos os geradores de números aleatórios são baseados no uso de fórmulas determinísticas precisas. Estas fórmulas fazem com que, a partir de um valor inicial chamado **semente**, seja gerada uma série de números aleatórios (pseudo-aleatórios). Em um determinado ponto da série, voltamos a semente e como a série é gerada por uma fórmula, a série, obviamente, se repete.
   A **quantidade de números gerados** até a seqüencia começar a se repetir é chamada de **Período**.
   Sempre desejamos o maior período possível. Para propósitos práticos o período deve ser, no mínimo, grande o suficiente para não se repetir durante a execução de uma simulação.

3. Eficiência Computacional

   Desde que um estudo de simulação necessita de que um grande número de variáveis aleatórias sejam geradas, o gerador de números aleatórios deve gerar estes números gastando o mínimo de tempo de computador. Além disto o gerador não deve usar muita memória. Com a evolução dos computadores esta última propriedade está perdendo um pouco de sua importância.

2.2 Métodos para a geração de números aleatórios

2.2.1 Método dos quadrados médios

Um dos primeiros métodos de geração de números aleatórios foi o chamado Método dos Quadrados Médios. Este método foi desenvolvido por John Von Neumann na década de 40. A técnica começa com um número inicial chamado de semente. O número é então elevado ao quadrado e os dígitos do meio do número gerado formam o próximo número da seqüência. Este segundo número é então elevado ao quadrado e os números do meio do número gerado são o próximo número da seqüência e assim por diante...

Exemplo: Gerar uma seqüência de números aleatórios de 4 dígitos. Seja 3187 a semente normalmente rotulada como x_0.

$x_0 = 3187$

$(3187)^2 = 10 \mid 1569 \mid 69 \Rightarrow x_1 = 1569$

$(1569)^2 = 02 \mid 4617 \mid 61 \Rightarrow x_2 = 4617$

$(4617)^2 = 21 \mid 3166 \mid 89 \Rightarrow x_3 = 3166$

$(3166)^2 = 10 \mid 0235 \mid 56 \Rightarrow x_4 = 235$

$(235)^2 = 00 \mid 0552 \mid 25 \Rightarrow x_5 = 552$

$(552)^2 = 00 \mid 3047 \mid 04 \Rightarrow x_6 = 3047$

e assim por diante...

Este método apresenta 2 problemas sérios: normalmente os períodos são curtos e se o n$^{\underline{o}}$ gerado é 0, o método só apresenta zero!

Exemplo: Gerar, pelo método dos quadrados médios, números pseudo aleatórios de 2 dígitos tendo 44 como semente.

$x_0 = 44$

$(44)^2 = 1 \mid 93 \mid 6 \Rightarrow x_1 = 93$

$(93)^2 = 8 \mid 64 \mid 9 \Rightarrow x_2 = 64$

$(64)^2 = 4 \mid 09 \mid 6 \Rightarrow x_3 = 9$

$(9)^2 = 0 \mid 08 \mid 1 \Rightarrow x_4 = 8$

$(8)^2 = 0 \mid 06 \mid 4 \Rightarrow x_5 = 6$

$(6)^2 = 0 \mid 03 \mid 6 \Rightarrow x_6 = 3$

$(3)^2 = 0 \mid 00 \mid 9 \Rightarrow x_7 = 0$

$(0)^2 = 0 \mid 00 \mid 0 \Rightarrow x_8 = 0$

2.2.2 Métodos Congruentes

A maioria dos métodos usados hoje em dia são variações do chamado Método Congruente Linear, cujos pontos básicos foram propostos por Lehmer em 1951. Neste método os números aleatórios, gerados sucessivamente, são obtidos da relação recursiva:

$$\boxed{x_{n+1} = (ax_n + c) \mod m}$$

A função $z \bmod t$ dá o resto da divisão **inteira** de z por t (ex. $23 \bmod 5 = 3$).
A constante **a** é chamada de multiplicador, a constante **c** é o incremento e **m** é o módulo. Como antes, x_0 é a semente.
Quando $c = 0$, o método é chamado de Congruência Multiplicativa.
O Método da Congruência Linear ($c \neq 0$), por gerar números aleatórios que tendem a ter mais dificuldades em passar nos testes estatísticos de aleatoriedade dos que os gerados pelo método da Congruência Multiplicativa ($c = 0$), não é, praticamente, mais usado hoje em dia.

Exemplo: Gerar números aleatórios, usando o método congruente multiplicativo, tendo os seguintes valores: $x_0 = 3$, $a = 2$ e $m = 10$.
$x_0 = 3$
$x_1 = (2 \times 3) \mod 10 = 6$
$x_2 = (2 \times 6) \mod 10 = 2$
$x_3 = (2 \times 2) \mod 10 = 4$
$x_4 = (2 \times 4) \mod 10 = 8$
$x_5 = (2 \times 8) \mod 10 = 6$
Como podemos observar o período desta geração foi muito curto (=4).
Ficou claro também, neste pequeno exemplo, que o número aleatório gerado é o resto da divisão **inteira** por m, ou seja um número inteiro entre 0 e $(m - 1)$.

> A fórmula congruente é necessária para se gerar números aleatórios, mas não suficiente. A seleção dos valores de $a, c,$ e m afeta drasticamente as propriedades estatísticas da geração bem como o tamanho do período.

2.3 Números Aleatórios uniformemente distribuídos em [0,1)

Como já explicado anteriormente, a fórmula congruente gera números aleatórios **inteiros** no intervalo $[0, m - 1)$.
Uma convenção estabelece que um **gerador de números aleatórios básico** deve gerar números no intervalo $[0, 1)$. Para conseguir isto, todo gerador de números aleatórios divide o número gerado por m. Desta forma o que se obtém é uma distribuição uniforme distribuída em [0,1).

Assim, por exemplo, para $a = 13$, $m = 67$ e $x_0 = 1$, teríamos:
$x_0 = 1 \div 67 = 0.0149253$
$x_1 = (13 \times 1) \bmod 67 = 13 \div 67 = 0.1940298$
$x_2 = (13 \times 13) \bmod 67 = 35 \div 67 = 0.522388$
$x_3 = (13 \times 35) \bmod 67 = 53 \div 67 = 0.7910447$
$\vdots$
Alguns geradores dividem por $m - 1$ o que dá uma distribuição $[0, 1]$. Na verdade como m é sempre um número muito grande, dividir por m ou $(m-1)$ é irrelevante.

2.4 O gerador RANDU e a formação de treliças

O método da congruência multiplicativa pode facilmente ser implementado em linguagens de programação como Pascal, Java, Basic, etc..., por exemplo.
Um gerador, chamado de RANDU, foi desenvolvido pela IBM e durante quase 20 anos foi usado por praticamente todos os modelos de simulação nas décadas de 60 e 70.
A RANDU utilizava os valores $a = 65.539$ e $m = 2^{31} = 2.147.483.648$, ou seja

$$x_{n+1} = (65539 \times x_n) \quad \bmod \ 2147483648$$

Assim considerando uma semente igual a 313, teríamos:
$X_0 = 313$
$X_1 = (65539 \times 313) \mod 2147483648 = 20513707 \div 2148483648 =$ **0,0095524**
$X_2 = (65539 \times 20513707) \mod 2147483648 = 123079425 \div 2148483648 =$ **0,0573133**
$X_3 = (65539 \times 123079425) \mod 2147483648 = 553853187 \div 2148483648 =$ **0,257907**
$\vdots$
A função, codificada em Pascal[1] é a seguinte: (a rotina original era em Fortran)

```
semente : longint;
Function RANDU: Double;
Const
  a = 65539;
  m = 2147483647;
  q = 32766;
  r = 32774;
Var
  lo, hi, test : longint;
Begin
  hi := semente div q;
  lo := semente mod q;
  test := a * lo - r * hi;
```

[1]Todos os programas listados a seguir foram desenvolvidos com a finalidade exclusiva de ilustrar a matéria apresentada. Não houve qualquer preocupação em otimizá-los ou usar as técnicas mais refinadas de programação.
Outro fator a ser considerado é que certos comandos usados nos programas podem não funcionar, dependendo do compilador Pascal que estiver sendo usado.

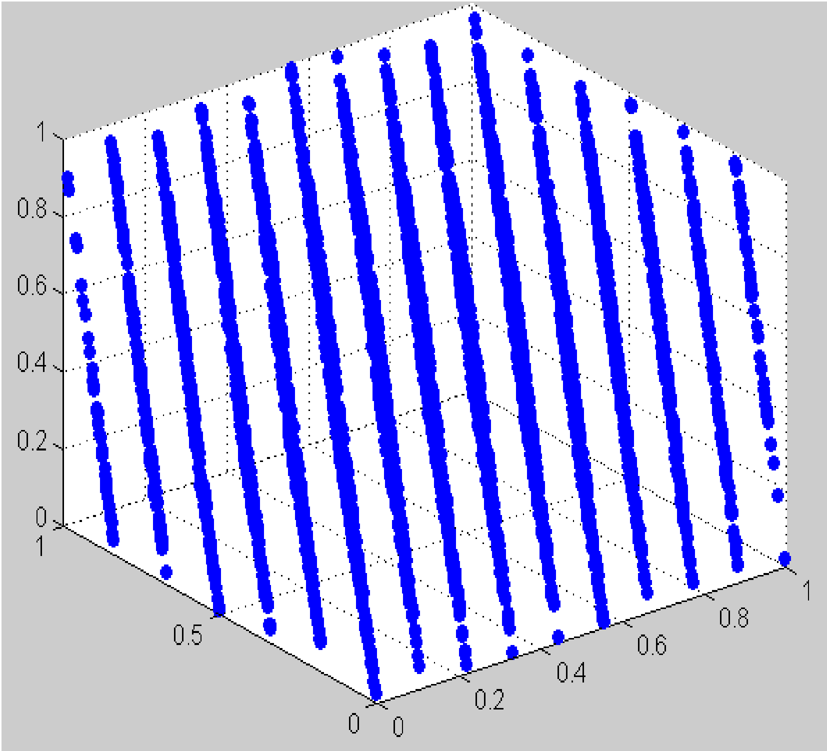
```
  If test >= 0 Then
    Begin
      semente := test;
      RANDU := semente / m
    End
  Else
    Begin
      semente := test + m;
      RANDU := semente / m;
    End;
End;
```

A seguir um programa Pascal que utiliza o gerador acima. O programa pede a semente, que deve ser um inteiro entre 1 e 32767, e imprime os 5 primeiros números aleatórios gerados.

```
{USO DA RANDU}
Var
  I: Integer;
  ALEAT: Double;
{$I RANDU.PAS}
Begin
    Writeln('Qual a semente ? (1 - 32767) ');
    Readln(semente);
    For I:=1 to 5 do
      Begin
        ALEAT := RANDU;
        Writeln(ALEAT:10:8);
      End;
End.
```

Para uma semente igual a 313, os números impressos por este programa são: 0.0095524, 0.0573133, 0.2579080, 0.0316280 e 0.8685962.

Na década de 70 diversos trabalhos provaram que a rotina RANDU apresentava resultados estatisticamente ruins.
Um dos problemas era a formação de treliças (lattice em inglês) quando se traçava gráficos com sucessivos números aleatórios gerados pela RANDU.
Vamos imaginar um cubo com lados igual a 1, ou seja variando de 0 a 1. Vamos marcar pontos neste cubo com coordenadas (x_1, x_2, x_3),(x_2, x_3, x_4) , (x_3, x_4, x_5), etc..., onde x_i é um número gerado pela RANDU a partir de uma semente qualquer.
Vamos marcar 5.000 pontos. O que deveríamos esperar ?
Que o cubo fosse preenchido uniformemente pelos pontos plotados. O que ocorre no entanto é que todos os pontos caem em 15 planos formando o que parece ser uma treliça (daí o nome). Nenhum ponto cai entre os planos, como podemos ver no gráfico a seguir:



Este aspecto, identificado claramente na RANDU, fez com que esta rotina fosse abandonada como geradora de números aleatórios.
A partir daí apareceram diversas alternativas que veremos a seguir.

2.5 O gerador RAND1

Foi apresentado pela pela IBM para substituir a RANDU e está baseado na relação $x_{i+1} = (16807 \times x_i) \mod 2147483647$, ou seja, $a = 16807$ e $m = 2^{31} - 1$.

O nome RAND1, assim como outros nomes que usaremos mais adiante, foram dados no sentido de facilitar o entendimento da matéria. Deve ficar claro, no entanto, que **exceto** a RANDU, nenhum outro gerador tem, na literatura técnica, nome próprio.

A rotina a seguir, é a sua implementação em Pascal.

```
semente : longint;
Function RAND1: Double;
Const
  a = 16807;
  m = 2147483647;
  q = 127773;
  r = 2836;
Var
  lo, hi, test : longint;
Begin
  hi := semente div q;
  lo := semente mod q;
  test := a * lo - r * hi;
  If test > 0 Then
    Begin
      semente := test;
      RAND1 := semente / m;
    End
  Else
    Begin
      semente := test + m;
      RAND1 := semente / m;
    End;
End;
```

O programa a seguir imprime os 5 primeiros números aleatórios gerados a partir de uma semente, que deve ser um inteiro entre 1 e 32767, escolhida.

```
{USO DA RAND1}
Var
  I: Integer;
  ALEAT: Double;
{$I RAND1.PAS}
Begin
    Writeln('Qual a semente ? (1 - 32767) ');
    Readln(semente);
    For I:=1 to 10 do
      Begin
        ALEAT := RAND1;
        Writeln(ALEAT:10:8);
      End;
End.
```

Para uma semente igual a 313 os 5 primeiros números gerados são: 0.0024496, 0.1713277, 0.5044658, 0.5574913 e 0.7561453.

2.6 O gerador RAND2

Este gerador está baseado no trabalho de Marse e Robert [Marse, K., and S. D. Roberts: Implementing a Portable Fortran Uniform (0,1) Generator, Simulation, 41: 135-139 (1983)] e tem sua base na relação
$x_{i+1} = (630360016 \times x_i) \mod 2147483647$,
ou seja $a = 630360016$ e $m = 2^{31} - 1$.
A seguir apresentamos a sua implementação em Pascal:

```
semente : double;
Function RAND2: Double;
Const
  MULT1 = 24112;
  MULT2 = 26143;
  B2E15 = 32768;
  B2E16 = 65536;
  MODLUS = 2147483647;
Var
  HI15,HI31,LOW15,LOWPRD, OVFLOW, g: Longint;
  Z,Z1: Double;
Begin
     g := trunc(semente);
     HI15 := g Div B2E16;
     LOWPRD := (g-HI15*B2E16)*MULT1;
     LOW15 := LOWPRD Div B2E16;
     HI31 := HI15*MULT1+LOW15;
     OVFLOW := HI31 Div B2E15;
     g := (((LOWPRD-LOW15*B2E16)-MODLUS)+(HI31-OVFLOW*B2E15)*B2E16)+OVFLOW;
     If g < 0 Then g := g + MODLUS;
     HI15 := g DIV B2E16;
     LOWPRD := (g-HI15*B2E16)*MULT2;
     LOW15 := LOWPRD Div B2E16;
     HI31 := HI15*MULT2+LOW15;
     OVFLOW := HI31 Div B2E15;
     g := (((LOWPRD-LOW15*B2E16)-MODLUS)+(HI31-OVFLOW*B2E15)*B2E16)+OVFLOW;
     If g < 0 Then g := g + MODLUS;
     semente := g;
     RAND2 := (2*(semente/256)+1)/16777216.0;
End;
```

O programa a seguir imprime os 5 primeiros números gerados pela RAND2. A semente que é pedida no início do programa, deve ser um inteiro entre 1 e 2147483646.

```
{USO DA RAND2}
Var
  I: Integer;
  ALEAT: Double;
{$I RAND2.PAS}
Begin
    Writeln('Qual a semente ? (1 - 2147483646) ');
    Readln(semente);
    For I:=1 to 6 do
      Begin
        ALEAT := RAND2;
        Writeln(ALEAT:10:8);
      End;
End.
```

Para uma semente igual a 20006270, os 5 primeiros números gerados são 0.592880, 0.436457, 0.556578, 0.617909 e 0.768997.

2.7 Métodos com períodos maiores

Os métodos vistos anteriormente ainda são muito usados, inclusive a RANDU!. No entanto, para simulações mais complexas, seus períodos são relativamente curtos (2 bilhões de números) o que levou ao desenvolvimento de novos métodos de geração de números aleatórios, todos com a característica comum de ter grandes períodos ($> 10^{30}$).

2.7.1 O gerador RAND3

Este Gerador está baseado no trabalho "Toward a Universal Random Number Generator – George Marsaglia, Florida State University Report: FSU-SCRI-87-50 (1987)".
Ele se enquadra em uma classe de geradores chamados de "Geradores de Fibonacci" e estão baseados na seguinte recorrência:

$$x_i = (x_{i-p} + x_{i-k}) \; mod \; m \quad \text{onde } p > k > 0$$

Obviamente que o método precisa gerar p números para se obter o primeiro número aproveitável. No caso do programa abaixo temos $p = 97$ e $k = 33$ e o método pede 2 sementes. A 1ª no intervalo [1, 31328] e a 2ª no intervalo [1, 30081].
Para cada par de sementes é gerada uma série de números aleatórios com período de 10^{30}.
Assim, variando-se as sementes temos 900.000.000 de séries, cada uma com 10^{30} números aleatórios.
A rotina em Pascal (RAND3) é a seguinte:

```
{RAND3}
Var
 semente1, semente2 : longint;
 I97,J33, Param : Longint;
 CR3,CDR3,CMR3 : Double;
 UVETOR: Array[1..98] of Double;
Function RAND3: Double;
VAR
  S,T,UNI: Double;
  J,K,L,II,JJ,M: Longint;
Begin
    If semente1 <> 0 Then
       Begin
         I97 := 97;
         J33 := 33;
         UNI := 0;
         semente1 := semente1 Mod 31329;
         semente2 := semente2 Mod 30082;
         Param := ((semente1 Div 177) Mod 177) + 2;
         J := (semente1 Mod 177) + 2;
         K := ((semente2 Div 169) Mod 178) + 1;
         L := semente2 Mod 169;
         For II := 1 to 97 do
          Begin
            S := 0.0;
            T := 0.5;
            For JJ := 1 to 24 do
             Begin
               M := ((Param*J) Mod 179)*K;
               M := M Mod 179;
               PARAM := J;
               J := K;
               K := M;
               L := (53*L+1) Mod 169;
               If ((L*M) Mod 64) >= 32 Then S := S + T;
               T := T * 0.5;
             End;
            UVETOR[II] := S;
          End;
         CR3 := 362436.0 / 16777216.0;
         CDR3 := 7654321.0 / 16777216.0;
         CMR3 := 16777213.0 / 16777216.0;
         RAND3 := UNI;
         Exit;
       End;
    UNI := UVETOR[I97] - UVETOR[J33];
    If UNI < 0.0 Then UNI := UNI + 1.0;
    UVETOR[I97] := UNI;
    I97 := I97 -1;
    If I97 = 0 Then I97 := 97;
    J33 := J33 -1;
    If J33 = 0 Then J33 := 97;
    CR3 := CR3 - CDR3;
```

```
    If CR3 < 0.0 Then CR3 := CR3 + CMR3;
    UNI := UNI - CR3;
    If UNI < 0 Then UNI := UNI + 1.0;
    RAND3 := UNI;
End;
```

A seguir temos um programa que usa a RAND3 e imprime os 5 primeiros números gerados a partir de um par de sementes.

```
{USO DA RAND3}
Var
  I : Integer;
  ALEAT : Double;
{$I RAND3.PAS}
Begin
    Writeln('Qual a 1a. SEMENTE ? (1 - 31328) ');
    Readln(semente1);
    Writeln('Qual a 2a. SEMENTE ? (1 - 30081) ');
    Readln(semente2);
    ALEAT:= RAND3;    {chamada inicial para passagem das sementes}
    semente1 := 0;    {obrigatorio antes de se usar a RAND3 abaixo}
    For I:=1 to 5 do                                      |
      Begin                                               |
        ALEAT := RAND3;    <<-----------------------------|
        Writeln(ALEAT:10:8);
      End;
End.
```

Podemos observar que é necessário uma chamada inicial da RAND3 para passar as sementes escolhidas. **Após esta chamada inicial, a variável semente1 deve ser feita igual a zero**. A partir daí, toda vez que se chamar, no programa, a RAND3, ela devolverá um número aleatório.
Com a 1ª semente igual a 1802 e a 2ª igual a 9373 os 5 primeiros números gerados são: 0.116391, 0.964849, 0.88297, 0.420487 e 0.495856.

2.7.2 O gerador RAND4

Um outro gerador, que chamamos de RAND4, tem um período ainda maior. A sua descrição pode ser vista no *paper* "An object-oriented random-number package with many long streams and substreams – Pierre L'Ecuyer, Richard Simard, E.Jack Chen and W. David Kelton - 2000".
Este gerador pertence a uma classe de geradores que nada mais são do que a combinação de 2 ou mais geradores congruentes, baseados na regra de que o período de 2 geradores "embaralhados" é maior do que de um gerador só. No caso deste

gerador, seu período é de 3.1×10^{57}. As fórmulas de geração são:

$$\begin{aligned}
A_i &= (1403580 \times A_{i-2} - 810728 \times A_{i-3}) mod (2^{32} - 209) \\
B_i &= (527612 \times B_{i-1} - 1370589 \times B_{i-3}) mod (2^{32} - 22853) \\
Y_i &= (A_i - B_i) mod (2^{32} - 209) \\
U_i &= \frac{Y_i}{2^{32} - 209}
\end{aligned}$$

A implementação do método permite que sejam geradas 10^{19} séries diferentes, cada uma delas com 10^{38} números aleatórios !!
A rotina em Pascal (RAND4) é a seguinte:

```
{RAND4.PAS}
Type
MATRIZ1 = Array[0..2,0..2] of Double;
VETOR3 = Array[1..50,0..5] of Double;
VETOR6 = Array[0..5] of Double;
Const
M1: Double = 4294967087.0;
M2: Double = 4294944443.0;
NORM: Double = 1.0 / (4294967087.0 + 1.0);
A12: Double = 1403580.0;
A13N: Double = 810728.0;
A21: Double  = 527612.0;
A23N: Double = 1370589.0;
TWO17: Double = 131072.0;
TWO53: Double = 9007199254740992.0;
A1P127: MATRIZ1 =  (
       (     2427906178.0, 3580155704.0,  949770784.0),
       (     226153695.0,  1230515664.0,  3580155704.0),
       (     1988835001.0, 986791581.0,   1230515664.0)
       );

A2P127: MATRIZ1 = (
       (     1464411153.0,  277697599.0, 1610723613.0 ),
       (       32183930.0, 1464411153.0, 1022607788.0 ),
       (     2824425944.0,   32183930.0, 2093834863.0 )
       );

Var
CG, BG, IG: VETOR3;
Semente:  VETOR6;
SERIE: Integer;
{-------------------------------------------------------------------------------}
Function MULTMODM (A, S, C, M: Double): Double;
Var
  V: Double;
  A1: Longint;
Begin
    V := A * S + C;
    If ((V >= TWO53) or (V <= -TWO53)) then
```

```
    Begin
        A1 := trunc(A / TWO17);
        A := A - (A1 * TWO17);
        V  := A1 * S;
        A1 := trunc(V / M);
        V := V - (A1 * M);
        V := V * TWO17 + A * S + C;
    End;
    A1 := trunc(V / M);
    V := V - (A1 * M);
    If (V < 0.0) then  MULTMODM := V + M   else MULTMODM := V;
End;
{-----------------------------------------------------------------------}
Procedure MATVECMODM (Var A: MATRIZ1; Var S: array of Double; Var V:
                                          Array of Double; M: Double);
Var
   I: Integer;
   X: Array[0..2] of Double;
   ZERO: Double;
Begin
    ZERO := 0.0;
    For I := 0 to 2 do
       Begin
        X[I] := MULTMODM (A[I][0], S[0], ZERO, M);
        X[I] := MULTMODM (A[I][1], S[1], X[I], M);
        X[I] := MULTMODM (A[I][2], S[2], X[I], M);
       End;
     For I := 0 to 2 do
       Begin
        V[I] := X[I];
       End;
End;
{-----------------------------------------------------------------------}
Function RAND4 : Double;
Var
   K: Longint;
   P1, P2, U: Double;
Begin
    { Componente 1 }
    P1 := A12 * CG[SERIE][1] - A13N * CG[SERIE][0];
    K := trunc(P1/M1);
    P1 := P1 - (K * M1);
    if (P1 < 0.0) then P1 := P1 + M1;
    CG[SERIE][0] := CG[SERIE][1];
    CG[SERIE][1] := CG[SERIE][2];
    CG[SERIE][2] := P1;
    { Componente 2 }
    P2 := A21 * CG[SERIE][5] - A23N * CG[SERIE][3];
    K := trunc(P2/M2);
    P2 := P2 - (K * M2);
    if (P2 < 0.0) then P2 := P2 + M2;
    CG[SERIE][3] := CG[SERIE][4];
    CG[SERIE][4] := CG[SERIE][5];
```

```
    CG[SERIE][5] := P2;
    { Combinacao}
    if (P1 > P2) then U := (P1 − P2) ∗ NORM
        else U := (P1 − P2 + M1) ∗ NORM;
    RAND4 := U;
End;
{——————————————————————————————————————————————}
Procedure PASSASEMENTE;
Var
   I,J: Integer;
   TEMP: array[0..2] of Double;
Begin
  For I := 0 to 5 do
      Begin
        BG[1][I] :=  Semente[I];
        CG[1][I] :=  Semente[I];
        IG[1][I] :=  Semente[I];
      End;
   For J := 2 to 50 do
     Begin
       For I := 0 to 2 do
          Begin
           TEMP[I] := Semente[I];
          End;
     MATVECMODM (A1P127, TEMP, TEMP, M1);
     For I := 0 to 2 do
       Begin
        Semente[I] := TEMP[I];
       End;
     For I := 0 to 2 do
       Begin
        TEMP[I] := Semente[I + 3];
       End;
     MATVECMODM (A2P127, TEMP, TEMP, M2);
     For I := 0 to 2 do
       Begin
        Semente[I + 3] := TEMP[I];
       End;
     For I := 0 to 5 do
       Begin
         BG[J][I] :=  Semente[I];
         CG[J][I] :=  Semente[I];
         IG[J][I] :=  Semente[I];
       End;
   End;
 End;
```

Esta versão do programa só permite que se use 50 séries, mas devemos ressaltar que isto equivale a se ter 50 geradores diferentes, cada um com 10^{38} números aleatórios !

Para a RAND4 são necessárias 6 sementes iniciais, todas no intervalo [1; 4294967087]. O programa a seguir, utiliza a RAND4 para gerar números aleatórios. A partir das 6 sementes informadas e da série escolhida, ele imprime os 5 primeiros números gerados.

```
{USORAND4}
Var
  I: Longint;
  ALEAT: Double;
{$I Rand4.pas}
Begin
  For I := 0 to 5 do
     Begin
       Writeln('Semente ',I+1,' [1 - 4294967087] ? ');
       Readln(semente[I]);
       If (Semente[I] < 1.0 ) or (Semente[I] > 4294967087.0) then
         Begin
           Writeln('Semente fora da faixa permitida !');
           Exit;
         End;
     End;
{=========================================================}
PASSASEMENTE;      {———>> Chamada obrigatoria antes de se usar a RAND4}
{=========================================================}
    Writeln('Informe a Serie a ser usada [1 - 50}');
    Readln(SERIE);
    For I:= 1 to 5 do
     Begin
       ALEAT := RAND4;
       Writeln(ALEAT:10:8);
     End;
 End.
```

Para as sementes 78975, 2731847, 1300, 15873476, 7590 e 6150 e série igual a 4 os números aleatórios encontrados foram: 0.57235, 0.50824, 0.89428, 0.94493 e 0.51779 .

2.8 Geradores de números aleatórios embutidos

Praticamente todas as linguagens de programação (Pascal, Java, Basic, C, etc...) tem comandos para gerar números aleatórios uniformemente distribuídos em [0, 1]. Os chamados programas aplicativos, como o Excel por exemplo, também tem, já programado, rotinas para gerar números aleatórios.

2.8.1 O gerador do Turbo Pascal

Para exemplificar podemos ver a seguir um programa em Pascal para gerar e imprimir números aleatórios.

```
{Gerador embutido do Turbo Pascal}
Var
  Semente,I: Integer;
  ALEAT: Real;
Begin
    Writeln('Qual a semente (1 - 32767) ? ');
    Readln(Semente);
    RandSeed := Semente;
    For I:=1 to 5 do
    Begin
      Writeln(Random:10:8);
    End;
End.
```

O Pascal tem, já predefinida, uma variável, chamada RandSeed, que é a semente para a rotina interna do Turbo Pascal para a geração de números aleatórios. Assim se fizermos RanSeed igual a um valor no intervalo [1 , 32767] estaremos fornecendo a semente para o gerador. Cada vez que a função Random é chamada, temos a geração de um número aleatório em [0, 1].
No programa acima, se escolhermos a semente igual a 45, os 5 números impressos são: 0.912097, 0.866463, 0.730010, 0.867894 e 0.715913.
O Pascal tem também uma instrução chamada Randomize que atribui a semente do gerador (variável RandSeed) um valor calculado a partir da hora corrente do computador no momento da execução do programa.
Vemos a seguir um exemplo de um programa utilizando o comando Randomize, observando que o Randomize deve vir logo no início do programa.

```
{Gerador embutido do Turbo Pascal}
{Uso do Randomize}
Var
  I: Integer;
  ALEAT: Real;
Begin
    Randomize;
    For I:=1 to 5 do
    Begin
      Writeln(Random:10:8);
    End;
End.
```

É óbvio que usando-se o Randomize, não conseguimos repetir a mesma seqüência de números gerados já que a semente muda em cada execução do programa.
O gerador do Turbo Pascal está baseado na seguinte fórmula congruente linear:
$x_{i+1} = (134775813 \times x_i + 1) \mod 4294967296$

2.8.2 O gerador do Excel

O uso de planilhas, principalmente o Excel que tem mais de 90% do mercado, é largamente utilizada na simulação de modelos de pequeno e médio porte. Desta forma, o gerador embutido do Excel tem sido objeto de muitos estudos de avaliação da sua qualidade "estatística".
Até a versão 2003 o Excel usava um gerador baseado na seguinte fórmula congruente: $x_{i+1} = (9821 \times x_i + 0.211327) \mod 1$
Usar "mod 1" é equivalente a se pegar a parte fracionária da conta $(9821 \times x_i + 0.211327)$.
Inúmeros trabalhos, já publicados mostram que este gerador, embora não tão ruim como a RANDU, também tem problemas de uniformidade e aleatoriedade.
A partir da versão 2003, ou seja no Excel 2003 e no Excel 2007, o gerador passou a ser o descrito no *paper* "Building a Random Number Generator – Wichman, B.A. and I.D. Hill, Byte, pp. 127 – 128, March - 1987".
Este gerador é a combinação de 3 geradores e tem período de 10^{13}. Suas fórmulas são:
$A_{i+1} = (171 \times A_i) \mod 30269$
$B_{i+1} = (172 \times B_i) \mod 30307$
$C_{i+1} = (170 \times C_i) \mod 30323$
$ALEAT = [(A_{i+1} \div 30269) + (B_{i+1} \div 30307) + (C_{i+1} \div 30323)] \mod 1$
Inúmeros trabalhos técnicos mostraram que este é um gerador de boa qualidade.

2.8.3 Cuidados a serem tomados

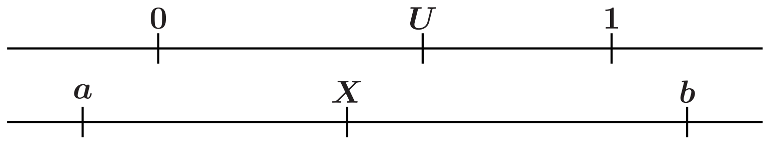
Quando usamos um gerador embutido, que em muitos casos não estão documentados, é importante se testar a qualidade do gerador. Se não houver tempo para que estes testes sejam realizados, deve-se tentar encontrar algum material já publicado sobre a qualidade do gerador em questão. Em hipótese alguma, devemos usar, em aplicações mais sensíveis, um gerador "desconhecido". Muitos geradores embutidos nada mais são do que a RANDU !!

2.9 Números aleatórios com outras distribuições uniformes

Uma vez que temos uma rotina para gerar números uniformemente distribuídos no intervalo $[0, 1]$, é fácil gerá-los com outras distribuições uniformes.

2.9.1 Variáveis aleatórias contínuas

Suponha que X é uma variável aleatória contínua, uniformemente distribuída dentro do intervalo (a, b), onde $a < b$. Seja U uma variável aleatória uniformemente distribuída no intervalo $[0, 1]$.



Aplicando proporcionalidade simples temos:

$$\left(\frac{X-a}{b-a}\right) = \left(\frac{U-0}{1-0}\right)$$

ou

$$X = a + (b - a) \times U$$

Assim é muito simples gerar X de um dado U, conhecendo-se a e b.
O programa Pascal a seguir gera e imprime os 5 primeiros números aleatórios uniformemente distribuídos no intervalo $[a, b]$.
O programa usa a RAND2 como gerador básico.

```
{Variaveis aleatorias [a - b]}
Var
  I: Integer;
  U, X, a, b: Double;
{$I RAND2.PAS}
Begin
    Writeln('Qual a semente ? (1 - 2147483646) ');
    Readln(semente);
    Writeln('Qual o limite inferior (a) ?');
    Readln(a);
    Writeln('Qual o limite superior (b) ?');
    Readln(b);
    For I:=1 to 5 do
      Begin
        U := RAND2;
        X := a + (b - a) * U;
        Writeln(X:10:5);
      End;
End.
```

Com uma semente da RAND2 igual a 7777, **a** igual a 1 e **b** igual a 4, o programa gera e imprime os seguintes números: 3.44774, 3.25192, 1.93325, 2.80820 e 3.37664.

2.9.2 Variáveis aleatórias discretas

Agora suponha que a e b sejam quantidades inteiras, $a < b$, e X é uma variável aleatória discreta (só valores inteiros) uniformemente distribuída no intervalo $[a, b]$. Assim X só pode tomar valores iguais a $a, a + 1, a + 2, ..., b - 1, b$.
Se U é contínua e uniformemente distribuída no intervalo $[0, 1]$, então:

$$X = a + \text{INT}[(b - a + 1) \times U]$$

onde INT é a função inteiro, ou seja a que elimina as decimais.
Observe que como $0 \leq U < 1$, a quantidade $\text{INT}\{(b-a+1) \times U\}$ toma os valores inteiros $0, 1, 2, .., (b - a)$. Logo X só pode tomar valores $a, a + 1, a + 2, ..., b$.
Vejamos um exemplo: Seja $a = 2$ e $b = 7$.
Assim X será igual a:
$X = 2 + \text{INT}[(7 - 2 + 1) \times U]$
$X = 2 + \text{INT}[6 \times U)] \quad \Rightarrow 6 \times U$ dá um valor entre 0 e 5.9999
$X = 2+$ um valor de 0 a 5, inteiro.
Logo X pode ser $2, 3, 4, 5, 6$ ou 7. O program a seguir simula a jogada de 2 dados e utiliza a RAND1 como gerador básico.

```
{Variaveis aleatorias Inteiras [a - b]}
{Simulacao de jogar 2 dados}
Var
  I, a , b, X1, X2: Integer;
  U: Double;
{$I RAND1.PAS}
Begin
    Writeln('Qual a semente ? (1 - 32767) ');
    Readln(semente);
    Writeln('Qual o limite inferior (a) ?');
    Readln(a);
    Writeln('Qual o limite superior (b) ?');
    Readln(b);
    For I:=1 to 5 do
      Begin
        U := RAND1;
        X1 := a + Trunc((b - a + 1) * U);
        U := RAND1;
        X2:= a + Trunc((b - a + 1) * U);
        Writeln(X1 + X2);
      End;
End.
```

Obs. Em Pascal o comando para pegar só a parte inteira de um número é Trunc.

Escolhendo uma semente igual a 875 para a RAND1, o programa gera e imprime os seguintes valores: 2, 3, 6, 2 e 9.

2.10 Testes estatísticos para a uniformidade

Existem numerosos testes estatísticos para garantir que os números pseudos-aleatórios estão sendo gerados aleatoriamente. Veremos alguns deles sem nos aprofundarmos na teoria estatística.
Um teste básico que sempre deve ser realizado para validar um gerador de números aleatórios é o teste de uniformidade. Veremos 2 dos mais usados: o teste de Kolmogorov-Smirnov e o teste do Qui-Quadrado (χ^2).
Ambos os testes medem o grau de aderência entre a distribuição de uma amostra de números aleatórios gerados e a distribuição uniforme teórica. Ambos os testes estão baseados na hipótese nula de que nenhuma diferença significante existe entre a amostra e a distribuição teórica.

2.10.1 O teste do χ^2 (qui-quadrado)

Chamando O_i de frequência observada e E_i de frequência esperada na categoria i, o χ^2 pode ser calculado da seguinte forma:

$$\chi^2_{calculado} = \frac{(O_1 - E_1)^2}{E_1} + \frac{(O_2 - E_2)^2}{E_2} + \ldots + \frac{(O_k - E_k)^2}{E_k}$$

Este teste é feito com um determinado nível de significância, α, normalmente 5%. O teste diz que se o χ^2_{calc} for menor que o $\chi^2_{tabelado}$ aceitamos a hipótese nula (H_0), ou seja de que a sequência de números aleatórios gerados é uniformemente distribuída em $[0, 1]$.
Vejamos um exemplo: sejam os números aleatórios a seguir, gerados por um gerador básico qualquer:

0.34	0.90	0.25	0.89	0.87	0.44	0.12	0.21	0.46	0.67
0.83	0.76	0.79	0.64	0.70	0.81	0.94	0.74	0.22	0.74
0.96	0.99	0.77	0.67	0.56	0.41	0.52	0.73	0.99	0.02
0.47	0.30	0.17	0.82	0.56	0.05	0.45	0.31	0.78	0.05
0.79	0.71	0.23	0.19	0.82	0.93	0.65	0.37	0.39	0.42
0.99	0.17	0.99	0.46	0.05	0.66	0.10	0.42	0.18	0.49
0.37	0.51	0.54	0.01	0.81	0.28	0.69	0.34	0.75	0.49
0.72	0.43	0.56	0.97	0.30	0.94	0.96	0.58	0.73	0.05
0.06	0.39	0.84	0.24	0.40	0.64	0.40	0.19	0.79	0.62
0.18	0.26	0.97	0.88	0.64	0.47	0.60	0.11	0.29	0.78

Dividindo em 10 intervalos [0 – 0.1), [0.1 – 0.2), [0.2 – 0.3), ... , [0.9 – 1] e contando a quantidade de números gerados em cada intervalo (O_i), podemos efetuar os cálculos para se testar com o χ^2. Como temos 100 números e 10 faixas iguais, o valor esperado (E_i) em cada faixa é igual a 10.

Intervalo	O_i	E_i	$O_i - E_i$	$(O_i - E_i)^2$	$(O_i - E_i)^2/E_i$
0.0 – 0.1	8	10	-2	4	0.4
0.1 – 0.2	8	10	-2	4	0.4
0.2 – 0.3	10	10	0	0	0.0
0.3 – 0.4	9	10	-1	1	0.1
0.4 – 0.5	12	10	2	4	0.4
0.5 – 0.6	8	10	-2	4	0.4
0.6 – 0.7	10	10	0	0	0.0
0.7 – 0.8	14	10	4	16	1.6
0.8 – 0.9	10	10	0	0	0.0
0.9 – 1.0	11	10	1	1	0.1
	100	100			$\sum = 3.4$

$\chi^2_{calc} = 3.4$
O valor do χ^2_{tab} é encontrado em tabelas onde 2 parâmetros são necessários: o número de graus de liberdade ν (nu) e o nível de significância (α).
O número de graus de liberdade é igual ao número de intervalos $-$ 1. O nível de significância será 5% ou 0.05.
Logo com $\nu = 10 - 1 = 9$ e $\alpha = 0.05$ encontramos:
$\chi^2_{tab} = 16.9$ (tabela página 147)
No nosso exemplo como $3.4 < 16.9$, nós aceitamos a hipótese de que os números foram gerados uniformemente distribuídos em $[0, 1]$.
O programa abaixo usa a RAND3 para gerar 5.000 números aleatórios e a seguir calcula o χ^2_{calc}.

```
{TESTE D0 QUI-QUADRADO}
Const
N = 5000;
Var
  I,J: Integer;
  ALEAT: Double;
  QUIC: Real;
  A: Array[1..5000] of Double;
  INTERV: Array[1..10] of Integer;
{$I RAND3.PAS}
Begin
     Writeln('Qual a 1a. SEMENTE ? (1 - 31328) ');
     Readln(semente1);
     Writeln('Qual a 2a. SEMENTE ? (1 - 30081) ');
     Readln(semente2);
     ALEAT:= RAND3;
     semente1 := 0;
{AMOSTRA DE N NUMEROS ALEATORIOS }
     For I:=1 to N do
       Begin
        ALEAT := RAND3;
        A[I] := ALEAT;
       End;
{SOMA OS NUMEROS NO SEU INTERVALO}
     For I:=1 to N do
```

```
      Begin
       A[I] := A[I] * 10;
       J := TRUNC(A[I]);
       INTERV[J+1] := INTERV[J+1] + 1;
      End;
    QUIC := 0.0;
    ALEAT := N / 10.0;
{CALCULA E IMPRIME O QUI-QUADRADO}
    For I := 1 to 10 do
      Begin
        QUIC := QUIC + SQR(INTERV[I] - ALEAT) / ALEAT;
      End;
    Writeln('QUI-QUADRADO Calculado = ',QUIC:10:5);
End.
```

Com as sementes 5432 e 7654 da RAND3, o valor encontrado (χ^2_{calc}) é de 5,26. Como este valor é menor que 16.9, aceitamos a hipótese de que a RAND3 gera números aleatórios uniformemente distribuídos.

2.10.2 O teste de Kolmogorov-Smirnov

Neste teste comparamos a distribuição acumulada teórica, $F(x)$, da distribuição uniforme com a distribuição acumulada, $S_N(x)$, para uma amostra de **N** valores gerados por um gerador básico (U_{is}). Por definição,

$$F(x) = P(X \leq x) = x, \quad 0 \leq x \leq 1$$

Se a amostra dos N números aleatórios gerados é $U_1, U_2, U_3, ..., U_N$, então a distribuição acumulada da amostra, $S_N(x)$, é definida por:

$$S_N(x) = \frac{\text{número de } U_{is} \text{ que são } \leq x}{N}$$

Se N é razoavelmente grande, $S_N(x)$ deve se tornar uma boa aproximação para $F(x)$.
O teste KS está baseado no valor absoluto da maior diferença entre $F(x)$ e $S_n(x)$, ou seja, está baseado na estatística:

$$D = \max \mid F(x) - S_N(x) \mid$$

A distribuição de D é conhecida e tabelada, podendo-se testá-la contra a distribuição acumulada uniforme. As etapas do teste são os seguintes:

I) Classifique os N números da amostra em ordem crescente de modo que $U_1 \leq U_2 \leq \ldots \leq U_N$.

II) Calcule para cada U_i da amostra

$$D^+ = \left\{\frac{i}{N} - U_i\right\}$$

$$D^- = \left\{U_i - \frac{i-1}{N}\right\}$$

Desconsidere os D^+ e D^- negativos.

III) Calcule: $D =$ o maior entre todos os D^+ e D^-

IV) Determine o valor crítico, $D\alpha$, de uma tabela para um nível de significância α e uma amostra de tamanho N.

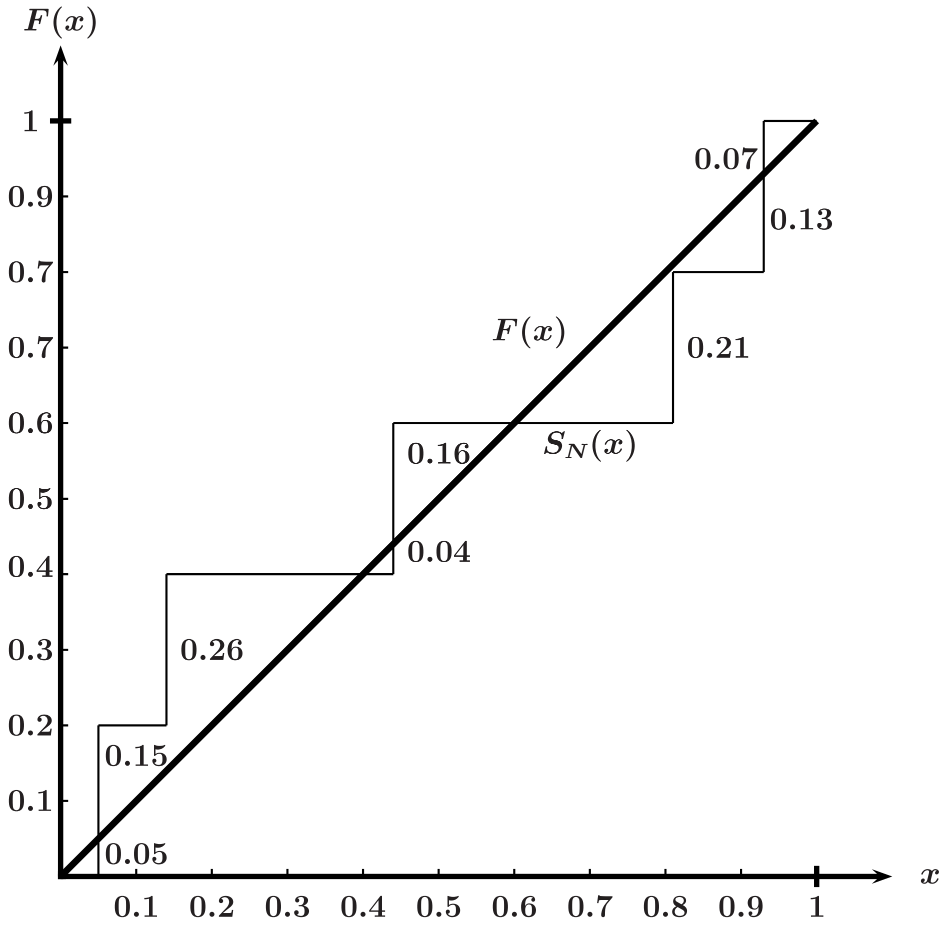
V) Se $D > D\alpha$ então a hipótese nula (os números aleatórios foram gerados uniformemente distribuídos) é rejeitada. Se $D \leq D\alpha$, então a hipótese é aceita.

Exemplo: Sejam 0.44, 0.81, 0.14, 0.05 e 0.93, cinco números aleatórios gerados para os quais desejamos usar o teste de KS, com um nível de significância α de 5%. A tabela a seguir mostra os cálculos necessários, lembrando que, inicialmente, os números devem ser classificados em ordem crescente:

i	1	2	3	4	5
U_i	0.05	0.14	0.44	0.81	0.93
$S(x) = \frac{i}{N}$	0.20	0.40	0.60	0.80	1.00
$D^+ = \frac{i}{N} - U_i$	0.15	0.26	0.16	–	0.07
$D^- = U_i - \frac{(i-1)}{N}$	0.05	–	0.04	0.21	0.13

Temos então:
$D = \max(0.15, 0.26, 0.16, 0.07, 0.05, 0.04, 0.21, 0.13) = 0.26$

O gráfico a seguir ilustra os cálculos da tabela acima podendo-se observar que 0.26 é a maior diferença entre a função acumulada teórica $F(x)$ e a função acumulada da amostra $S_N(x)$.



Podemos consultar uma tabela ou usar uma **aproximação** que diz que aceitamos a hipótese nula (a distribuição é uniformemente distribuída em $[0, 1]$) se:

$$\left(\sqrt{N} + 0.12 + \frac{0.11}{\sqrt{N}}\right) \times D \leq C_\alpha$$

Para $\alpha = 0.05$, o valor de $C_{0.05}$ é 1.358. Como, no nosso exemplo, $N = 5$, temos:

$$\left(\sqrt{5} + 0.12 + \frac{0.11}{\sqrt{5}}\right) \times 0.26 = 0.625$$

Como 0.625 é menor que 1.358, aceitamos a hipótese, ou seja, ou números foram gerados uniformemente distribuídos em $[0, 1]$.

O programa a seguir calcula o valor de D para uma amostra de 5.000 números aleatórios gerados pela RAND2.

```
{TESTE DE KOLMOGOROV-SMIRNOV}
Type
Vetor = Array[1..5000]of Double;
Var
  I: Integer;
  D,U: Double;
  N,DIFE,DIFD: Real;
  A :Vetor;
{$I RAND2.PAS}
{*************************************************}
Procedure CLASSIFICA(LEFT, RIGHT: Integer; Var A: Vetor);
Var
  II,L1,R1,I,J: Integer;
  K,TR: Double;
Begin
    K := (A[LEFT] + A[RIGHT])/2.;
    I := LEFT;
    J := RIGHT;
    Repeat
    While A[I] < K do
          INC(I,1);
    While K < A[J] do
          DEC(J,1);
    If I <= J then
          Begin
          TR := A[I];
          A[I] := A[J];
          A[J] := TR;
          INC(I,1);
          DEC(J,1);
          End;
    Until I > J;
    If LEFT <J then
          CLASSIFICA(LEFT,J,A);
    If I < RIGHT then
          CLASSIFICA(I,RIGHT,A);
    End;
{*************************************************}
Begin
    N := 5000;
    Writeln('Qual a semente ? (1 - 2147483646) ');
    Readln(semente);
    For I :=1 to Trunc(N) do
      Begin
       U := RAND2;
       A[I] := U;
      End;
    CLASSIFICA(1,Trunc(N),A);
    D := -9999.0;
    For I := 1 to Trunc(N) do
      Begin
       DIFD := (I / N) - A[I];
       If DIFD < 0 then DIFD := -9999;
```

```
        DIFE := A[I] - ((I-1) / N);
        If DIFE < 0 then DIFE := -9999;
        If DIFE > D then D := DIFE;
        If DIFD > D then D := DIFD;
      End;
    Writeln('D = ',D:10:7);
End.
```

A execução deste programa, com a semente 656544, vai imprimir o valor de 0,0131642 para D.
Temos então:
$$\left(\sqrt{5000} + 0.12 + \frac{0.11}{\sqrt{5000}}\right) \times 0.0131642 = 0.9324$$
Como 0.9324 é menor que 1.358, aceitamos que a RAND2 gera uma sequência de números aleatórios uniformemente distribuídos em $[0, 1]$.

Convém neste ponto esclarecer a diferença entre uniformidade e aleatoriedade ou independência. Assim a seqüência 0.01, 0.10, 0.15, 0.20, 0.25, etc..., é uniforme mas não é aleatória. **Uniformidade é necessária mas não é suficiente**.

2.11 Testes de Aleatoriedade (Independência)

2.11.1 O teste do Intervalo (Gap test)

O teste do intervalo é usado para testar a ordem de uma seqüência de dígitos aleatórios. Neste teste nós contamos o número de dígitos entre a $iésima$ e a $iésima+1$ ocorrência de um dígito particular d. Se n outros dígitos estão presentes temos então um intervalo (gap) de comprimento n. Todos os intervalos são determinados para cada d, onde d pode ser um dos 10 valores possíveis (0,1,...,9). Tabulamos então o total do número de ocorrências de cada comprimento de intervalo.
O número esperado de intervalos de comprimento n é dado por :
$E_n = (0.9)^n(0.1)K$, onde K é número total de intervalos.
Podemos então usar o teste do χ^2 para verificar a aleatoriedade.
No caso de números gerados por um gerador básico, ou seja uniforme em $[0, 1]$, temos que relacionar intervalos com os dígitos de 0 a 9.
Assim sendo, se o número estiver no intervalo $[0, 0.1)$, consideramos como sendo o número 0. Se estiver no intervalo $[0.1, 0.2)$, consideramos como sendo igual a 1 e assim por diante. Exemplo: Sejam os 48 números aleatórios a seguir:

0.956	0.283	0.365	0.056	0.604	0.013
0.447	0.934	0.788	0.766	0.132	0.039
0.832	0.097	0.719	0.471	0.074	0.652
0.894	0.532	0.256	0.684	0.313	0.248
0.917	0.546	0.357	0.936	0.045	0.369
0,576	0.632	0.248	0.561	0.245	0.611
0.377	0.501	0.872	0.727	0.506	0.622
0.175	0.973	0.540	0.492	0.717	0.809

Associando os números gerados em cada faixa aos dígitos correspondentes, encontramos a seguinte seqüencia:
923 060 497 710 807 406 852 632 953 903 562 526 358 756 195 478.
Usando o número 0 como exemplo, vamos encontrar os intervalos (gap's):
O 1º intervalo tem comprimento 1 pois, na seqüencia, temos o 1º zero, logo depois o número 6 e, a seguir, outro 0. O próximo 0 aparece depois dos números 47171, ou seja temos um intervalo de comprimento igual a 5. A seguir aparecem os números 7 e 4 e outro 0. Temos então um intervalo de comprimento 2. Finalmente o último 0 vai aparecer depois dos números 68526329539. Temos um intervalo de comprimento igual a 11.
Procedendo de forma idêntica para os outros dígitos (1 a 9), podemos construir a tabela a seguir:

d	Comprimento Intervalo (n)
0	1, 5, 1, 2, 11
1	31
2	18, 2, 8, 1
3	19, 3, 2, 6
4	8, 29
5	5, 4, 2, 3, 2, 3
6	12, 3, 9, 3, 5
7	0, 4, 24, 6
8	5, 19, 8
9	6, 16, 2, 15

Um total de 38 intervalos foram encontrados, variando de 0 a 31 (o maior intervalo possível é 46).
O número esperado e observado de intervalos de cada comprimento estão tabulados a seguir:

n	O_n	E_n	n	O_n	E_n	n	O_n	E_n
0	1	3.80	17	0	0.63	34	0	0.11
1	3	3.42	18	1	0.57	35	0	0.10
2	6	3.08	19	2	0.51	36	0	0.09
3	5	2.77	20	0	0.46	37	0	0.08
4	2	2.49	21	0	0.42	38	0	0.07
5	4	2.24	22	0	0.37	39	0	0.06
6	3	2.02	23	0	0.34	40	0	0.06
7	0	1.82	24	1	0.30	41	0	0.05
8	3	1.64	25	0	0.27	42	0	0.05
9	1	1.47	26	0	0.25	43	0	0.04
10	0	1.32	27	0	0.22	44	0	0.04
11	1	1.19	28	0	0.20	45	0	0.03
12	1	1.07	29	1	0.18	46	0	0.03
13	0	0.97	30	0	0.16			
14	0	0.87	31	1	0.14			
15	1	0.78	32	0	0.13			
16	1	0.70	33	0	0.12			

E_n foi calculado pela fórmula $E_n = (0.9)^n(0.1)K$, com K = 38.
Por exemplo, para $n = 3$ temos: $E_3 = (0.9)^3 \times (0.1) \times 38 = 2.77$
De maneira a aplicar o teste do χ^2 devemos combinar as categorias de modo que $E_i > 5$, para cada nova categoria. Isto é um pré-requisito para o uso do χ^2.
Assim obtemos:

i	n	O_i	E_i
1	0 – 1	4	7.22
2	2 – 3	11	5.85
3	4 – 6	9	6.75
4	7 – 10	4	6.25
5	11 – 16	4	5.58
6	17 – 46	6	6.08

Podemos calcular o χ^2.

$$\chi^2_{calc} = \frac{(4-7.22)^2}{7.22} + \frac{(11-5.85)^2}{5.85} + \ldots + \frac{(6-6.08)^2}{6.08} = 7.98$$

Para achar o χ^2_{tab} temos $\nu = 6 - 1 = 5$ e $\alpha = 0.05$.
Da tabela tiramos $\chi^2_{tab} = 11.07$.
Como χ^2_{calc} é menor que o χ^2_{tab} aceitamos a hipótese que a seqüência foi aleatoriamente gerada.

2.11.2 O teste da corrida (Run test)

Uma corrida é uma sucessão de eventos similares, precedido e seguido por eventos diferentes. No nosso caso uma corrida será uma sucessão de números aleatórios crescentes ou decrescentes.
O procedimento consiste em contar o número total de corridas crescentes e decrescentes de tamanho n, onde $n = 1, 2, 3, ...$
O número observado de corridas pode então ser comparado com o número esperado de corridas. O valor esperado de corridas de tamanho n, se temos K números aleatórios, pode ser calculado de:

$$E_n = \begin{cases} \dfrac{2[(n^2+3n+1)K-(n^3+3n^2-n-4)]}{(n+3)!} & \text{para } n = 1,2,3,...,K-2 \\[2ex] \dfrac{2}{K!} & \text{para } n = K-1 \end{cases}$$

A aleatoriedade pode ser determinada pelo teste do χ^2.
Exemplo: Vamos aplicar o teste da corrida crescente e decrescente para os 50 números aleatórios apresentados a seguir, considerando que os números foram gerados da esquerda para a direita, por linha:

0.923	0.060	0.497	0.710	0.807
	−	+	+	+
0.406	0.852	0.632	0.953	0.903
−	+	−	+	−
0.562	0.526	0.358	0.756	0.195
−	−	−	+	−
0.478	0.262	0.318	0.720	0.837
+	−	+	+	+
0.017	0.513	0.199	0.083	0.335
−	+	−	−	+
0.233	0.035	0.116	0.848	0.432
−	−	+	+	−
0.133	0.451	0.081	0.467	0.249
−	+	−	+	−
0.299	0.575	0.600	0.695	0.380
+	+	+	+	−
0.979	0.849	0.999	0.892	0.580
+	−	+	−	−
0.544	0.378	0.872	0.427	0.548
−	−	+	−	+

O 1º número gerado foi 0,923. O 2º foi 0,060, logo houve uma diminuição e colocamos o sinal − embaixo de 0,060. A seguir foi gerado o número 0,497, ou seja houve uma subida. Colocamos o sinal + embaixo de 0,497. O próximo a ser gerado foi 0,710. Continuamos subindo e colocamos + embaixo deste último número. A seguir vem 0,807, ou seja subindo ainda e por isso, colocamos + embaixo de 0,807.

Aparece a seguir o número 0,406. Houve uma descida e portanto colocamos $-$ embaixo de 0,406.
Prosseguimos com este procedimento até o último número gerado.
Uma sucessão de sinais $+$ indica uma corrida crescente e uma sucessão de sinais $-$, indica uma decrescente.
Como temos 50 números, o tamanho das corridas pode variar de 1 a 49. Os tamanhos encontrados (O_n), assim como os valores esperados (E_n), calculados das fórmulas acima, estão mostrados na tabela a seguir:

n	O_n	E_n
1	23	20.92
2	4	8.93
3	2	2.51
4	3	0.53
5 – 49	0	0.11

Considerando que $K = 50$ o cálculo de E_2, por exemplo, é:

$$E_2 = \frac{2[(2^2 + 3 \times 2 + 1) \times 50 - (2^3 + 3 \times 2^2 - 2 - 4]}{(2+3)!} = 8.93$$

Reagrupando os dados de modo que $E_i > 5$ para cada categoria, temos:

i	n	O_i	E_i
1	1	23	20.92
2	2 – 49	9	12.08

Podemos calcular agora o χ^2:

$$\chi^2_{calc} = \frac{(23 - 20.92)^2}{20.92} + \frac{(9 - 12.08)^2}{12.08} = 0.99$$

O χ^2_{tab} para $\nu = 1$ e $\alpha = 0.05$ é igual a 3.841

Como o χ^2_{calc} é menor que o χ^2_{tab}, concluímos que os números aleatórios foram gerados aleatoriamente.

2.12 Observações finais sobre a geração de números aleatórios

Existe uma infinidade de testes estatísticos para testar tanto a aleatoriedade como a uniformidade. Vimos apenas alguns deles.
O registro importante que se deve fazer é que a confiabilidade do gerador de número aleatórios é a condição básica para se garantir a "qualidade" estatística necessária em um estudo de simulação.
Um gerador, não estatisticamente confiável, pode levar uma simulação a resultados desastrosos.
Também devemos considerar que testes mais rigorosos, quando não se conhece a qualidade do gerador a ser usado, devem ser feitos.
Um exemplo disto poderia ser a sequência de números mostrados a seguir:

0.123	0.060	0.497	0.710	0.807
0.406	0.952	0.632	0.953	0.903
0.562	0.526	0.358	0.956	0.195
0.478	0.262	0.318	0.720	0.837
0.917	0.513	0.199	0.083	0.335
0.233	0.035	0.936	0.848	0.432
0.133	0.451	0.081	0.467	0.949
0.299	0.575	0.600	0.695	0.380
0.979	0.912	0.312	0.892	0.580
0.544	0.378	0.872	0.927	0.548

É provável que uma seqüencia deste tipo passe em todos os testes que vimos anteriormente. No entanto esta seqüencia tem um sério desvio: de 7 em 7 números ela apresenta um número maior que 0.900!

Capítulo 3

Alguns modelos elementares de Simulação

A maioria dos problemas estudados através de modelos de simulação que acontecem na vida real necessitam do uso de variáveis aleatórias não uniformemente distribuídas.
No entanto, como ilustração, vamos ver 2 exemplos de simulação em que só é necessário ter um gerador de números aleatórios uniformemente distribuídos.
Nos 2 exemplos a seguir a subrotina RAND que aparece nos fluxogramas é uma subrotina genérica podendo ser a RAND1, a RAND2, uma RAND embutida de linguagem, etc...

3.1 Jogo de Dados ("Craps Game")

Um jogo popular nos E.Unidos é o chamado "craps" no qual 2 dados são jogados simultaneamente. Se a soma for 7 ou 11 você ganha. Se a soma for 2, 3 ou 12 você perde. Se der 4, 5, 6, 8, 9 ou 10 você tem direito a jogar os dados outra vez. Se a soma for 7 você perde. Se repetir o que deu na 1ª jogada você ganha. Se não, você pode jogar os dados outra vez e assim por diante.
A probabilidade de se ganhar neste jogo é de 49,3% obtida da teoria das probabilidades.
O programa a seguir é a implementação do jogo. Ele permite que a cada execução se escolha quantas vezes queremos jogar.
O programa usa o gerador interno do Turbo Pascal para gerar números aleatórios.
Podemos observar que o programa tem 2 "procedures". A 1ª, chamada <u>SUB1</u>, verifica à cada jogada dos dados se ganhamos, perdemos ou se temos que jogar os dados uma outra vez. A 2ª, chamada de <u>SUB2</u>, simula a jogada simultânea de 2 dados. Cada vez que ganhamos, o programa soma 1 a variável <u>SOMA</u>. No final dos N jogos, o programa imprime a divisão de SOMA por N, ou seja, o percentual de jogos ganhos.

```
{CRAPS   GAME}
Var
  JOGADA,GUARDA,RESULTADO:  Integer;
  N,  I  :  longint;
  SOMA:  Real;
{**************************************************}
Procedure SUB1 (Var JOGADA: Integer);
Begin
    JOGADA := TRUNC((1.0+6.0*Random))+
                              TRUNC((1.0+6.0*Random));
End;
{**************************************************}
PROCEDURE SUB2 (Var RESULTADO: Integer);
Begin
    SUB1(JOGADA);
    If (JOGADA = 7) or (JOGADA = 11) then
       Begin
         RESULTADO := 1;
         EXIT;
       End;
    If (JOGADA = 2) or (JOGADA = 3) or (JOGADA = 12) then
       Begin
         RESULTADO := 0;
         EXIT;
       End;
    GUARDA := JOGADA;
    RESULTADO := -9;
    Repeat
        SUB1(JOGADA);
        If JOGADA = 7 then
           Begin
             RESULTADO := 0;
             EXIT;
           End;
        If JOGADA = GUARDA then
           Begin
             RESULTADO := 1;
             EXIT;
           End;
   Until (RESULTADO = 0) or (RESULTADO = 1);
   EXIT;
End;
{**************************************************}
Begin
    Writeln('Qual a Semente (0 a 32767) ?');
    Readln(RandSeed);
    Writeln('Quantos Jogos ?');
    Readln(N);
    SOMA := 0.0;
    For I := 1 to N do
      Begin
        SUB2(RESULTADO);
        SOMA := SOMA + RESULTADO;
      End;
    Writeln((SOMA / N)*100:5:2,'%');
End.
```

A execução deste programa 5 vezes, cada vez com 10.000 jogos, apresentou os seguintes resultados:

Semente	Ganho (%)
8744	48,26
123	49,42
10000	49,21
7361	48,97
313	49,33
Média	**49,17**

3.2 Avaliação de Integrais

Uma aplicação interessante da simulação é o cálculo de integrais definidas, no sentido em que um método probabilístico é usado para resolver um problema determinístico. Este procedimento é chamado de **Método de Monte Carlo** e não necessita mais do que um gerador de números aleatórios uniformemente distribuídos em $[0, 1]$.
Suponha que se deseja calcular a integral:

$$I = \int_a^b f(x)dx$$

onde $f(x)$ representa uma curva contínua no intervalo $a \leq x \leq b$. Para uso do método, precisamos conhecer também o valor máximo (F_{max}) da função no intervalo (a, b).
As etapas do método de Monte Carlo são:

1. Gerar um nº aleatório uniformemente distribuído, U_x, cujo valor está entre a e b.

2. Calcular $f(U_x)$.

3. Gerar um 2º número aleatório, U_y, cujo valor está entre 0 e F_{max}. Estes 2 números aleatórios (U_x e U_y) representam as coordenadas de um ponto no espaço.

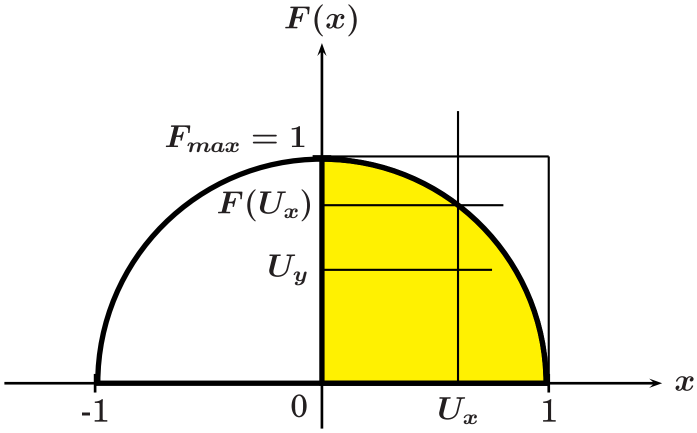
4. Comparar U_y com $f(U_x)$. Se $U_y \leq f(U_x)$ então o ponto (U_x, U_y) cairá em cima ou abaixo da curva, ou seja dentro da área que representa a integral.

5. Repita n vezes as etapas de 1 a 4, acumulando os pontos que caíram dentro da área da integral.

6. Calcule o percentual ($PERC$) de pontos que caíram na área da integral, dividindo pelo total de números tentados (n).

O valor da integral é obtido por:
$I = PERC \times (b - a) \times F_{max}$

Exemplo: Considere a integral abaixo:
$\int_0^1 \sqrt{1-x^2}dx$
O valor desta integral pode ser obtida pelo cálculo e é igual 0.785398. A usaremos para mostrar que a simulação produz resultados consistentes.
O gráfico da função e da "integral" pode ser visto a seguir:



A área da parte do gráfico abaixo da curva, no 1º quadrante, é o valor da integral procurada.
Neste exemplo a é 0 e b é 1. O F_{max} é 1.
A área do retângulo é igual a $(b - a) \times F_{max}$, ou seja $1 \times 1 = 1$. A integral, ou seja, a área abaixo da curva será igual a $(b - a) \times F_{max} \times \%$ de pontos que caem em cima ou abaixo da curva.
O programa a seguir implementa o método de Monte Carlo para cálculo de integrais definidas. Ele usa a RAND2 como gerador básico e tem uma função (VFUNCAO) onde a função a ser integrada, $\sqrt{1-x^2}$ no nosso exemplo, é colocada.

```
{CALCULO DE INTEGRAIS}
VAR
  N,I: Integer;
  U: Double;
  X,FMAX,LIMINF,LIMSUP,UX,UY,FUX,IDENTRO,INTEGRAL: Real;
{$I RAND2.PAS}
{***************************************************}
Function VFUNCAO (X: Real): Real;
Begin
   VFUNCAO := SQRT(1-X*X);
End;
{***************************************************}
Begin
     Writeln('Qual a semente ? (1 - 2147483646) ');
     Readln(semente);
     Writeln('Quantos Numeros ?');
     Readln(N);
     Writeln('Qual o valor de FMAX ?');
     Readln(FMAX);
     Writeln('Qual o LIMITE INFERIOR INTEGRACAO (a) ?');
```

```
    Readln(LIMINF);
    Writeln('Qual o LIMITE SUPERIOR INTEGRACAO (b) ?');
    Readln(LIMSUP);
    If (LIMINF >= LIMSUP) Then
      Begin
        Writeln('ERRO NOS LIMITES ');
        Exit;
      End;
    IDENTRO := 0;
    For I := 1 to N do
      Begin
        U := RAND2;
        UX := LIMINF + (U * (LIMSUP - LIMINF));
        U := RAND2;
        UY := U * FMAX;
        FUX := VFUNCAO(UX);
        If UY <= FUX then IDENTRO := IDENTRO +1;
      End;
   INTEGRAL := (IDENTRO / N)*(LIMSUP-LIMINF)*FMAX;
   Writeln('VALOR DA INTEGRAL = ',INTEGRAL:10:5);
End.
```

Executando-se este programa, com a semente 555666 para a RAND2 e escolhendo-se 10.000 números, o valor da integral é dado pelo programa como igual a 0.7848. Na verdade o Método de Monte Carlo é mais um dos métodos numéricos para a solução de integrais definidas que não tem solução analítica.

A integral usada no último exemplo é apropriada para se reforçar a regra básica do uso da simulação: Problemas que tem solução analítica nunca devem ser resolvidos por meio da simulação. Soluções analíticas darão sempre respostas "mais exatas" que as respostas fornecidas pela simulação.
Quando, no entanto, não se tem solução analítica, a simulação pode dar respostas bastante aproximadas.

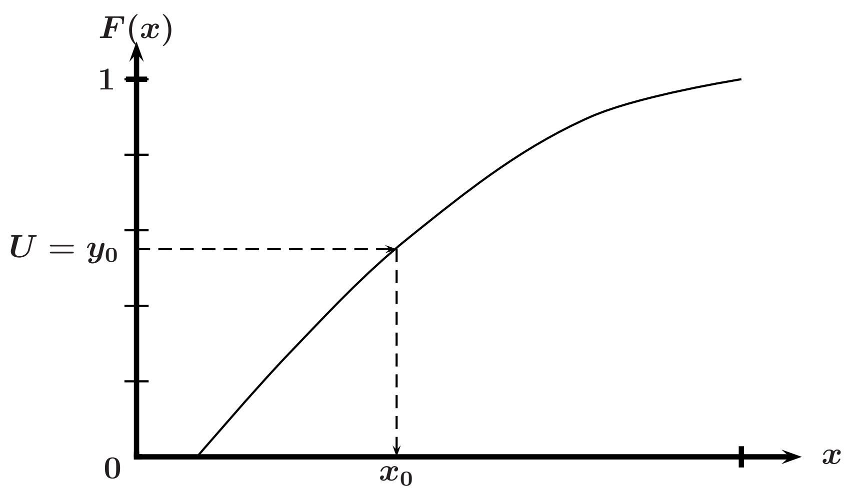
Capítulo 4

Variáveis aleatórias não uniformes

Na maioria dos problemas do mundo real, as variáveis aleatórias seguem distribuições diferentes da uniforme tal como a de Poisson, a Exponencial, a Normal, etc...
Neste capítulo veremos como números aleatórios uniformemente distribuídos em $[0, 1]$ podem ser usados para gerar variáveis aleatórias, não uniformes, como as citadas acima.

4.1 O Método da Transformação Inversa

Suponha que temos uma distribuição probabilística, com função de densidade $f(x)$ e função de distribuição acumulada igual a $F(x)$. Desejamos gerar uma variável aleatória que siga esta distribuição probabilística. O método da Transformação Inversa oferece uma maneira simples de resolver o problema.
O método está baseado no fato de que a distribuição acumulada, $F(x)$, tem valor entre 0 e 1 ou seja, no mesmo intervalo de um nº aleatório, U, gerado por um gerador básico. Assim sendo, tendo U, consideramos este valor como sendo um valor de $F(x)$. Para achar o valor de x correspondente, basta resolver a equação, ou seja, achar a inversa de $F(x)$.
A figura a seguir mostra, graficamente, o princípio no qual o método está baseado:



Isto é, se $y_0 = F(x_0)$, então podemos escrever:

$$x_0 = F^{-1}(y_0)$$

Substituindo y_0 por U, temos:

$$x_0 = F^{-1}(U)$$

Exemplo: Aplique o método da transformação inversa para a seguinte função de densidade probabilística:

$f(x) = \dfrac{x}{4}$ em $1 \leq x \leq 3$

$f(x) = 0$ fora do intervalo acima

Use o método para gerar 6 valores sucessivos para X, dados os seguintes valores aleatórios uniformemente distribuídos em $[0,1]$: 0.35, 0.97, 0.22, 0.15, 0.60, 0.43.

Inicialmente determinamos a função cumulativa:

$$y = F(x) = \int_1^x \left(\frac{x}{4}\right) dx = \left[\frac{x^2}{8}\right]_1^x = \frac{x^2}{8} - \frac{1}{8} = \frac{(x^2-1)}{8} \qquad \text{para } 1 \leq x \leq 3.$$

Resolvendo para x obtemos:

$x = \sqrt{8y+1}$

que pode ser escrita como:

$x_i = \sqrt{8U_i + 1}$

onde U_i é uma variável aleatória uniformemente distribuída em $[0,1]$.

Quando U_i = 0.35 nós podemos obter o valor correspondente para x_i como:

$x_i = \sqrt{(8) \times (0.350) + 1} = 1.95$

Procedendo de forma idêntica, podemos obter os demais valores:

i	U_i	x_i
1	0.35	1.95
2	0.97	2.96
3	0.22	1.66
4	0.15	1.48
5	0.60	2.41
6	0.43	2.11

O valor esperado e o desvio padrão podem ser calculados de:

$$E(x) = \int_{-\infty}^{+\infty} xf(x)dx = \int_1^3 x \times \frac{x}{4}dx = \int_1^3 \frac{x^2}{4}dx = \left[\frac{x^3}{12}\right]_1^3 = 2.167$$

$$\sigma^2 = \int_{-\infty}^{+\infty} x^2 f(x) - E(x)^2 = \int_1^3 x^2\frac{x}{4} - 2.167^2 = \left[\frac{x^4}{16}\right]_1^3 - 2.167^2 = 0.30555$$

$\sigma = \sqrt{0.30555} = 0.552$

Veremos a seguir como aplicar o método da transformação inversa para várias distribuições bem conhecidas.

Entretanto é bom esclarecer que este método não pode ser aplicado para todas as distribuições. Existem algumas, como a normal, cuja função de distribuição probabilística não pode ser integrada analíticamente e, logicamente, não se pode achar a sua inversa. Há casos ainda em que não é possível obter uma equação explícita para x mesmo se tendo uma expressão analítica para a função cumulativa. Existem outras técnicas para estes casos.

4.1.1 Distribuições Empíricas

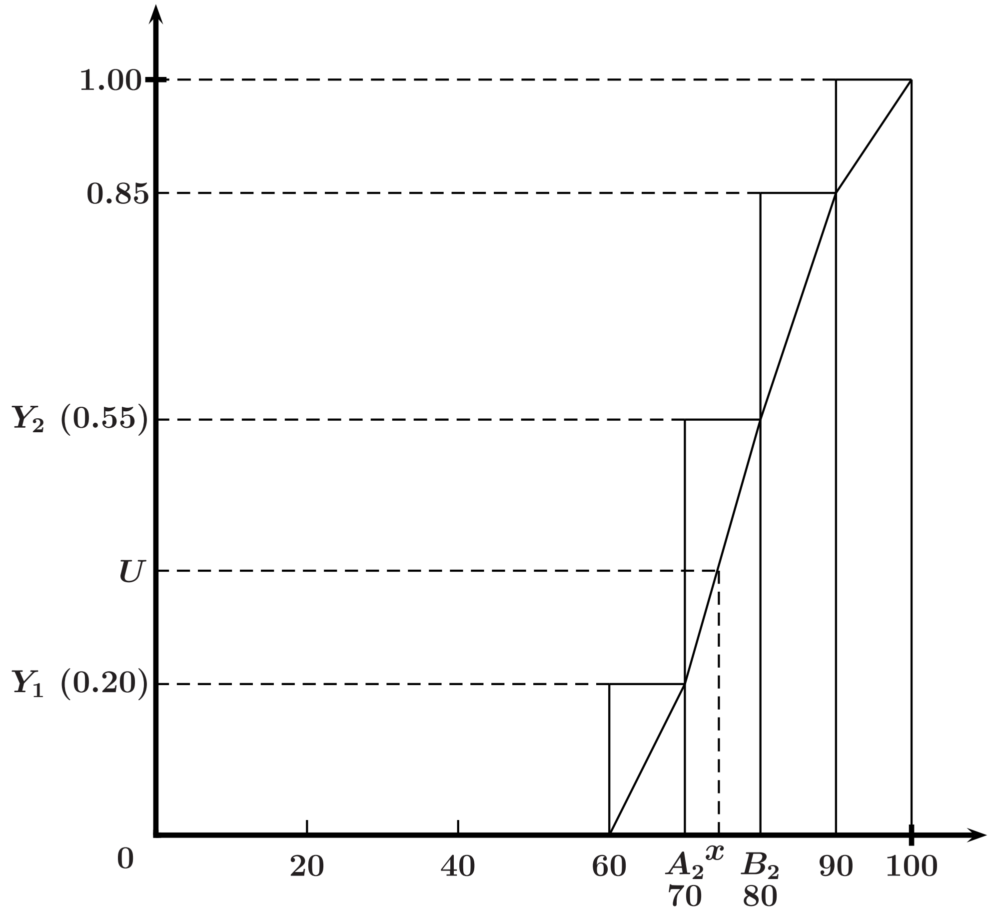
Em muitos problemas reais a probabilidade de que um evento ocorra é expressa em termos de dados empíricos agrupados. Vejamos um exemplo.
A seguinte distribuição empírica descreve o valor possível do custo para um novo produto a ser desenvolvido em determinada indústria:

Custo de Produção (em $ por unidade)	Probabilidade de ocorrência
60 – 70	0.20
70 – 80	0.35
80 – 90	0.30
90 – 100	0.15

Gerar 6 valores aleatórios para o custo de produção usando os seguintes números aleatórios uniformemente distribuídos (U_i): 0.35, 0.97, 0.22, 0.15, 0.60, 0.43.
Chamando de A_j o limite inferior e B_j, o limite superior, podemos construir a tabela da distribuição acumulada (Y_j):

j	A_j	B_j	Y_j
1	$60	$70	0.20
2	70	80	0.55
3	80	90	0.85
4	90	100	1.00

Cada valor de j indica a faixa da distribuição acumulada em que o valor de U se enquadra. Assim $j = 1$ compreende os valores de 0 a 0,20. Para $j = 2$ temos a faixa de maior que 0,20 a 0,55 e assim por diante.
Para obtermos uma fórmula de geração dos valores da distribuição vejamos o gráfico a seguir, onde queremos obter um valor x, a partir de um determinado valor U. O valor U se encontra na 2ª faixa da distribuição acumulada:



Usando interpolação linear e semelhança de triângulos, temos:

$$\frac{x - A_2}{B_2 - A_2} = \frac{U - Y_1}{Y_2 - Y_1}$$

Podemos tirar então o valor de X:

$$x = A_2 + \left[\frac{U - Y_1}{Y_2 - Y_1}\right][B_2 - A_2] \quad \text{ou generalizando,}$$

$$x = A_j + \left[\frac{U - Y_{j-1}}{Y_j - Y_{j-1}}\right][B_j - A_j]$$

O nosso 1º número aleatório, 0.35, cai na 2ª faixa ($j = 2$), porque é maior que 0.20 e menor que 0.55. Assim temos:

$$x = A_2 + \left[\frac{U - Y_1}{Y_2 - Y_1}\right][B_2 - A_2] = 70 + \frac{0.35 - 0.20}{0.55 - 0.20}[80 - 70] = 74.29$$

De maneira semelhante podemos calcular os outros números, obtendo:

j	U_i	X_i
1	0.35	\$74.29
2	0.97	98.00
3	0.22	70.57
4	0.15	67.50
5	0.60	81.67
6	0.43	76.57

Vamos ver os cálculos para $U = 0.15$ que cai na 1ª faixa, ou seja, $j = 1$:

$$x = A_1 + \left[\frac{U - Y_0}{Y_1 - Y_0}\right][B_1 - A_1]$$

Como Y_0 não existe, ou seja, é igual a zero, ficamos com:

$$x = A_1 + \left[\frac{U}{Y_1}\right][B_1 - A_1]$$

Substituindo pelos valores numéricos temos:

$$x = 60 + \left[\frac{0.15}{0.20}\right][70 - 60] = 67.50$$

4.1.2 A Distribuição Exponencial

Muitos modelos de simulação necessitam usar a distribuição exponencial. Isto é verdadeiro nos problemas que envolvem chegadas e partidas (filas), como a simulação de uma agência bancária, da saída (caixas) de um supermercado, de um aeroporto, etc...
A função de densidade probabilística da exponencial é igual a:

$$f(x) = \alpha e^{-\alpha x}$$

onde α é uma constante conhecida e a média (μ) é igual a $\frac{1}{\alpha}$.
A função de distribuição acumulada é dada por:

$$F(x) = 1 - e^{-\alpha x}$$

De modo a se fazer uso do método da transformação inversa devemos resolver para x. Assim temos:

$$x = -\frac{1}{\alpha}\ln[1 - F(x)]$$

Como a função acumulada, $F(x)$, é uniformemente distribuída em $[0, 1]$, a quantidade $1 - F(x)$ também será uniformemente distribuída em $[0, 1]$. Assim podemos escrever

$$x = -\left(\frac{1}{\alpha}\right)\ln(U)$$

onde x é a variável aleatória exponencialmente distribuída e U é um número aleatório uniformemente distribuído em $[0, 1]$, gerado por um gerador básico.
É muito comum que se precise gerar variáveis exponenciais x maiores ou iguais a um determinado valor positivo, x_0 , isto é $0 < x_0 < x$.
A equação acima fica:

$$x = x_0 - \left(\frac{1}{\alpha}\right)\ln(U)$$

A média (μ) fica como:

$$\mu = x_0 + \frac{1}{\alpha}$$

Podemos tirar então a relação entre α e μ torna-se:

$$\alpha = \frac{1}{(\mu - x_0)}$$

Exemplo: Gerar 6 variáveis exponencialmente distribuídas maiores que 2 e com média igual a 6, usando os seguintes números uniformemente distribuídos em $[0, 1]$: 0.35, 0.97, 0.22, 0.15, 0.60, 0.43. Temos então: $x_0 = 2$ e $\mu = 6$.
Calculando o valor de α:
$\alpha = \dfrac{1}{(6-2)} = 0.25$
Podemos então calcular as variáveis da distribuição exponencial:
$X_1 = 2 - \left(\dfrac{1}{0.25}\right)\ln(0.35) = 6.20$
$X_2 = 2 - \left(\dfrac{1}{0.25}\right)\ln(0.97) = 2.12$
Os demais valores encontrados são:

i	U_i	X_i
1	0.35	6.20
2	0.97	2.12
3	0.22	8.06
4	0.15	9.59
5	0.60	4.04
6	0.43	5.38

O programa a seguir, escrito em Pascal, implementa a geração de números aleatórios que seguem a distribuição exponencial.

O programa permite que se informe quantos números queremos gerar, o limite inferior da distribuição e a média.
Como saída ele imprime os 5 primeiros números gerados e a média de todos os números gerados.
O programa usa a RAND2 como o gerador básico de números aleatórios.

```
{EXPONENCIAL}
Var
  N, I ,NUMIMP: Integer;
  U,SOMA: Double;
  LIMINF,ALFA,MEDIA: Real;
{$I RAND2.PAS}
{*************************************************}
Function EXPONENCIAL (ALFA,LIMINF: Real; U: Double): Double;
Begin
    EXPONENCIAL := LIMINF - (1/ALFA) * LN(U);
End;
{*************************************************}
Begin
    Writeln('Qual a semente ? (1 - 2147483646) ');
    Readln(semente);
    Writeln('Quantos Numeros ?');
    Readln(N);
    Writeln('Qual o LIMITE INFERIOR ?');
    Readln(LIMINF);
    Writeln('Qual a Media ?');
    Readln(MEDIA);
    ALFA := 1.0 / (MEDIA - LIMINF);
    SOMA := 0.0;
    NUMIMP := 0;
    For I := 1 to N do
      Begin
        U := RAND2;
        U := EXPONENCIAL(ALFA,LIMINF,U);
        NUMIMP := NUMIMP + 1;
        If NUMIMP < 6 then Writeln(U:10:8);
        SOMA := SOMA + U;
      End;
   Writeln('MEDIA = ',SOMA / N:10:6);
End.
```

Em uma execução, usando-se a semente 1234 para a RAND2, em que foram gerados 10.000 números com limite inferior igual a 2 e média igual a 6, os seguintes resultados foram apresentados:
Primeiros 5 números gerados: 8.033; 10.615; 13.382; 7.210 e 2.457.
Média dos 10.000 números gerados: 6.00481.

4.1.3 A Distribuição Geométrica

Vamos agora considerar uma sequência de experiências independentes, onde o resultado de cada experiência é um sucesso ou um insucesso. Suponha que p é a probabilidade de sucesso de cada experiência ($0 < p < 1$), e $q = (1-p)$ é a probabilidade de insucesso. A probabilidade de x insucessos até um sucesso é dada por:

$$f(x) = pq^x$$

onde x deve ser um inteiro não negativo, isto é $x = 1, 2, \ldots$
A equação acima representa a densidade probabilística para a distribuição geométrica, cuja média é dada por:

$$\mu = \frac{q}{p}$$

e cuja variância é dada por:

$$\sigma^2 = \frac{q}{p^2} = \frac{\mu}{p}$$

A distribuição cumulativa correspondente pode ser expressa por:

$$\begin{aligned} F(x) &= f(0) + f(1) + f(2) + \ldots + f(x) \\ &= p + pq + pq^2 + \ldots + pq^x \\ &= \sum_{k=0}^{x} pq^k \end{aligned}$$

A função de distribuição cumulativa, $F(x)$, da distribuição geométrica varia no intervalo $[p, 1]$. A razão disto é porque $f(0) = p$.
De maneira a se usar o método da transformação inversa nós devemos expressar a distribuição cumulativa de uma forma um pouco diferente.
Para fazer isto observamos que:

$$\begin{aligned} \text{Prob}(X > 0) &= 1 - F(0) = 1 - p = q \\ \text{Prob}(X > 1) &= 1 - F(0) - F(1) = 1 - p - pq = 1 - p - p(1-p) = 1 - p - p + p^2 \\ &= 1 - 2p + p^2 \\ &= (1-p)^2 \\ &= q^2 \\ &\vdots \\ \text{Prob}(X > x) &= 1 - F(x) = q^{x+1} \end{aligned}$$

onde X é uma variável aleatória geométricamente distribuída.

Dividindo ambos os lados por q, temos:

$$\frac{1 - F(x)}{q} = q^x$$

Nós sabemos que $F(x)$ é uniformemente distribuída dentro do intervalo $[p, 1]$.
A diferença $1 - F(x)$ será uniformemente distribuída em $[1 - p, 0]$ ou $[0, 1 - p]$ ou $[0, q]$.
Logo a função $\dfrac{1 - F(x)}{q}$ será uniformemente distribuída sobre o intervalo [0,1].
Assim sendo, podemos escrever:

$$U = q^x$$

onde U é uma variável aleatória uniformemente distribuída no intervalo $[0, 1]$. Resolvendo para x obtemos:

$$x = \text{Inteiro}\left\{\frac{\ln U}{\ln q}\right\}$$

onde x será o valor da distribuição geométrica.

Exemplo: Gerar 6 variáveis aleatórias, geométricamente distribuídas, para um processo cuja probabilidade de sucesso é de 30%. Use os seguintes números aleatórios uniformemente distribuídos em $[0, 1]$: 0.35, 0.97, 0.22, 0.15, 0.60, 0.43.
Como $p = 0.3$ nós sabemos que $q = 1 - p = 0.7$. Logo podemos usar:
$X_1 = \text{Inteiro}(\ln 0.35 / \ln 0.70) = 2$
$X_2 = \text{Inteiro}(\ln 0.97 / \ln 0.70) = 0$
Os 6 números gerados são:

U_i	X_i
0.35	2
0.97	0
0.22	4
0.15	5
0.60	1
0.43	2

O programa a seguir, escrito em Pascal, implementa a geração de números aleatórios que seguem a distribuição geométrica.
O programa permite que se informe quantos números queremos gerar e a probabilidade de sucesso.
Como saída ele imprime os 5 primeiros números gerados, a média e o desvio padrão, calculado pela fórmula $\sqrt{(x_i^2)/n - \bar{x}^2}$, dos números gerados.
O programa usa a RAND2 como o gerador básico de números aleatórios.

```
{GEOMETRICA}
Var
  N, I ,NUMIMP, J : Integer;
  U,SOMA: Double;
  SUCESSO,INSUCESSO,MEDIA,DESVIO: Real;
{$I RAND2.PAS}
{*************************************************}
Function GEOMETRICA (INSUCESSO: Real; U: Double): Integer;
Begin
    GEOMETRICA := TRUNC(LN(U) / LN(INSUCESSO));
End;
{*************************************************}
Begin
    Writeln('Qual a semente ? (1 - 2147483646) ');
    Readln(semente);
    Writeln('Quantos Numeros ?');
    Readln(N);
    Writeln('Qual a probabilidade de sucesso ?');
    Readln(SUCESSO);
    INSUCESSO := 1 - SUCESSO;
    SOMA := 0.0;
    DESVIO := 0.0;
    NUMIMP := 0;
    For I := 1 to N do
      Begin
        U := RAND2;
        J := GEOMETRICA(INSUCESSO,U);
        NUMIMP := NUMIMP + 1;
        If NUMIMP < 6 then Writeln(J);
        SOMA := SOMA + J;
        DESVIO := DESVIO + J * J;
      End;
   Writeln('MEDIA = ',SOMA / N:10:5);
   Writeln('DESVIO PADRAO = ',SQRT((DESVIO/N)-(SOMA/N)*(SOMA/N)):10:5);
End.
```

Em uma execução, usando-se a semente 1234 para a RAND2, em que foram gerados 10.000 números com probabilidade de sucesso igual a 0.30, ou seja com média e desvio padrão teóricos iguais a:

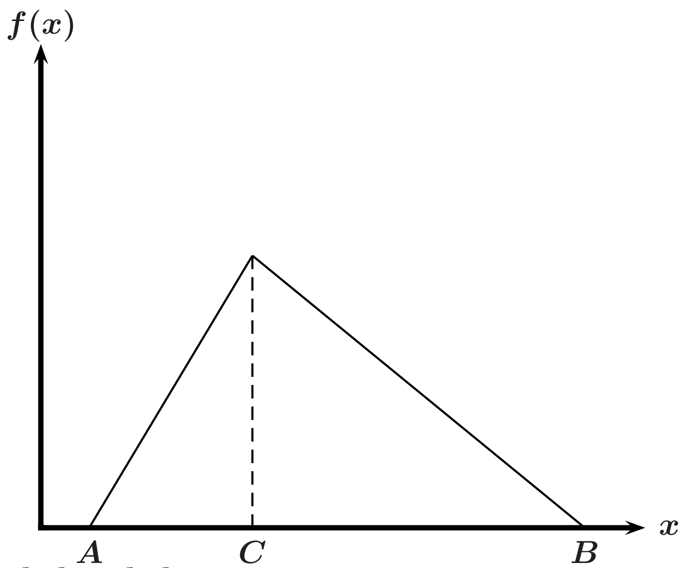
$$\boldsymbol{\mu = \frac{q}{p} = \frac{1 - 0.30}{0.30} = 2.3333}$$

$$\boldsymbol{\sigma = \sqrt{\frac{\mu}{p}} = \sqrt{\frac{2,3333}{0,30}} = 2.7886}$$

os seguintes resultados foram obtidos:
Primeiros 5 números gerados: 4, 6, 7, 3 e 0.
A média dos 10.000 números gerados foi igual a 2.3374.
O desvio padrão obtido foi igual a 2.7591.

4.1.4 A Distribuição Triangular

A distribuição Triangular tem um uso bastante difundido em Simulação, principalmente quando os dados disponíveis são poucos ou mesmo inexistentes.
Sua forma permite que dados não conclusivos sejam a ela adaptados, e seus limites, ou seja **A** (limite inferior), **C** (moda) e **B** (limite superior) sejam interpretados como os parâmetros mais pessimista (B), mais provável (C) e mais otimista (A) de uma determinada variável, como pode ser visto na figura a seguir.



Sua função de densidade é dada por:

$$\begin{aligned} f(x) &= \frac{2(x-A)}{(C-A)(B-A)} \quad \text{para } A \leq x \leq C \\ &= \frac{2(B-x)}{(B-C)(B-A)} \quad \text{para } C \leq x \leq B \end{aligned}$$

Podemos usar o Método da Transformação Inversa mas a descontinuidade da função faz com que seja necessário desenvolver 2 geradores separados: um para $x \leq C$ e um para $x \geq C$. Inicialmente vamos desenvolver um para $x < C$.
A função de distribuição acumulada $F(x)$ é igual a:

$$\begin{aligned} F(x) &= \int_A^x \frac{2(x-A)}{(C-A)(B-A)}\, dx \\ &= \frac{x^2 - 2Ax + A^2}{(C-A)(B-A)} \end{aligned}$$

Como $F(x)$ e U gerado por um gerador básico, variam em [0 , 1], podemos escrever:

$$U = \frac{x^2 - 2Ax + A^2}{(C-A)(B-A)}$$

Fazendo as simplificações necessárias, chegamos a :

$$x = A + \sqrt{U(C-A)(B-A)}$$

Esta fórmula pode ser usada se $U < \dfrac{(C-A)}{(B-A)}$, pois a razão $\dfrac{(C-A)}{(B-A)}$ é proporcional a área sob o triângulo de $x = A$ até $x = C$.
Para $x \geq C$ temos:

$$\begin{aligned} F(x) &= U = \int_C^x \frac{2(B-x)}{(B-C)(B-A)}\,dx \\ U &= \frac{-x^2 + 2Bx - B^2}{(B-C)(B-A)} + 1 \end{aligned}$$

Operando, para obter o valor de x, obtemos:

$$x = B - \sqrt{(1-U)(B-C)(B-A)}$$

Esta fórmula deve ser usada quando $U \geq \dfrac{(C-A)}{(B-A)}$.
A media e o desvio padrão da Distribuição Triangular são:

$$\begin{aligned} \overline{x} &= \frac{A+B+C}{3} \\ \sigma &= \sqrt{\frac{A^2+B^2+C^2-AB-AC-BC}{18}} \end{aligned}$$

Exemplo: Gerar 3 variáveis aleatórias seguindo a Distribuição Triangular com limite inferior (A) igual a 2, moda (C) igual a 4 e limite superior (B) igual a 8. Use os seguintes números aleatórios uniformemente distribuídos em $[0,1]$: 0.35, 0.97, 0.22.
A fórmula a ser usada depende se o valor de U é maior ou menor que $\dfrac{(C-A)}{(B-A)}$ ou seja $\dfrac{(4-2)}{(8-2)} = 0.3333$. Assim para o 1º U, 0.35 (maior que 0.333) temos:
$X_1 = 8 - \sqrt{(1-0.35)(8-4)(8-2)} = 4.050$
Para 0.97 (maior que 0.333) temos:
$X_2 = 8 - \sqrt{(1-0.97)(8-4)(8-2)} = 7.151$
Para 0.22 (menor que 0.333) temos:
$X_3 = 2 + \sqrt{0.22(4-2)(8-2)} = 3.624$
O programa a seguir, escrito em Pascal, implementa a geração de números aleatórios que seguem a distribuição triangular.
O programa permite que se informe quantos números queremos gerar e os 3 parâmetros (A, B e C) da distribuição.
Como saída ele imprime os 5 primeiros números gerados, a média e o desvio padrão dos números gerados.
O programa usa a RAND2 como o gerador básico de números aleatórios.

```
{TRIANGULAR}
Var
  N,I,NUMIMP: Integer;
  U,SOMA: Double;
  A,B,C,MEDIA,DESVIO,T: Real;
{$I RAND2.PAS}
{***********************************************}
Function TRIANGULAR (A, B, C : Real;U : Double): Real;
Begin
  If (U < ((C-A)/(B-A))) then
   TRIANGULAR := A + Sqrt(U*(C-A)*(B-A))
    Else TRIANGULAR := B - Sqrt((1-U)*(B-C)*(B-A));
End;
{***********************************************}
BEGIN
     Writeln('Qual a semente ? (1 - 2147483646) ');
     Readln(semente);
     Writeln('Quantos Numeros ?');
     Readln(N);
     Writeln('Qual o limite inferior (A) ?');
     Readln(A);
     Writeln('Qual o limite superior (B) ?');
     Readln(B);
     Writeln('Qual a moda (C) ?');
     Readln(C);
     SOMA := 0.0;
     DESVIO := 0.0;
     NUMIMP := 0;
     For I := 1 to N do
       Begin
         U := RAND2;
         T := TRIANGULAR(A,B,C,U);
         NUMIMP := NUMIMP + 1;
         If NUMIMP < 6 then Writeln(T:10:5);
         SOMA := SOMA + T;
         DESVIO := DESVIO + T * T;
       End;
    Writeln('MEDIA = ',SOMA / N:10:5);
    Writeln('DESVIO PADRAO = ',SQRT((DESVIO/N)-(SOMA/N)*(SOMA/N)):10:5);
END.
```

Em uma execução, usando-se a semente1234 para a RAND2, em que foram gerados 10.000 números com os parâmetros $A = 2$, $B = 8$ e $C = 4$ da triangular, os seguintes resultados foram obtidos:
Primeiros 5 números gerados: 3.629, 3.179, 2.835, 3.805 e 6.389.
A média dos 10.000 números gerados foi igual a 4.6621. A média teórica é igual a 4.66666.
O desvio padrão obtido foi igual a 1.248. O desvio padrão teórico é igual a 1.247.

4.1.5 A Distribuição de Weibull

A Distribuição de Weibull tem larga aplicação na área industrial pois inúmeros resultados, principalmente sobre a de vida útil de um componente, tem mostrado que variáveis que medem este tipo de resultado se ajustam muito bem a esta distribuição teórica.
Ela tem 2 parâmetros: α que reflete o tamanho da unidade na qual a variável aleatória x é medida e β que dá a forma (*shape*) da distribuição.
Sua função de densidade acumulada $F(x)$ é igual a:

$$F(x) = 1 - e^{-\left(\frac{x}{\alpha}\right)^{\beta}} \qquad \text{para } x, \alpha, \beta > 0$$

Sua média e desvio padrão são:

$$\begin{aligned} \overline{x} &= \alpha\Gamma\left(\frac{1}{\beta}+1\right) \\ \sigma &= \sqrt{\alpha^2\left[\Gamma\left(\frac{2}{\beta}+1\right) - \left[\Gamma\left(\frac{1}{\beta}+1\right)\right]^2\right]} \end{aligned}$$

Podemos usar a transformação inversa para gerar variáveis aleatórias seguindo a Distribuição de Weibull. Temos então:

$$U = F(x) = 1 - e^{-\left(\frac{x}{\alpha}\right)^{\beta}}$$

Resolvendo para x em função de U, chegamos a :

$$x = \alpha(-\ln U)^{1/\beta}$$

Exemplo: Gerar 6 variáveis aleatórias seguindo a Distribuição de Weibull com $\alpha = 4$ e $\beta = 1$. Use os seguintes números aleatórios uniformemente distribuídos em $[0, 1]$: 0.35, 0.97, 0.22, 0.15, 0.60, 0.43.
Temos então:
$X_1 = 4 \times (-\ln 0.35)^{1/1} = 4.1993$
$X_2 = 4 \times (-\ln 0.97)^{1/1} = 0.1218$
$X_3 = 4 \times (-\ln 0.22)^{1/1} = 6.0565$
$X_4 = 4 \times (-\ln 0.15)^{1/1} = 7.5885$
$X_5 = 4 \times (-\ln 0.60)^{1/1} = 2.0433$
$X_6 = 4 \times (-\ln 0.43)^{1/1} = 3.3759$
O programa a seguir, escrito em Pascal, implementa a geração de números aleatórios que seguem a Distribuição de Weibull.
O programa permite que se informe quantos números queremos gerar e os 2 parâmetros (α e β) da distribuição.
Como saída, ele imprime os 5 primeiros números gerados, a média e o desvio padrão dos números gerados.
O programa usa a RAND2 como o gerador básico de números aleatórios.

```
{WEIBULL}
Var
  N, I ,NUMIMP: Integer;
  U,SOMA: Double;
  ALFA,BETA,MEDIA,DESVIO: Real;
{$I RAND2.PAS}
{************************************************}
Function WEIBULL (ALFA, BETA: Real;U: Double): Real;
Begin
   WEIBULL := ALFA * EXP((1/BETA)* LN(-LN(U)));
End;
{************************************************}
Begin
    Writeln('Qual a semente ? (1 - 2147483646) ');
    Readln(semente);
    Writeln('Quantos Numeros ?');
    Readln(N);
    Writeln('Qual o valor de Alfa ?');
    Readln(ALFA);
    Writeln('Qual o valor de Beta ?');
    Readln(BETA);
    SOMA := 0.0;
    NUMIMP := 0;
    DESVIO := 0.0;
    For I := 1 to N do
      Begin
        U := RAND2;
        U := WEIBULL(ALFA,BETA,U);
        SOMA := SOMA + U;
        DESVIO := DESVIO + U * U;
        NUMIMP := NUMIMP + 1;
        If NUMIMP < 6 then Writeln(U:10:5);
      End;
   Writeln('MEDIA = ',SOMA / N:10:5);
   Writeln('DESVIO PADRAO = ',SQRT((DESVIO/N)-(SOMA/N)*(SOMA/N)):10:5);
End.
```

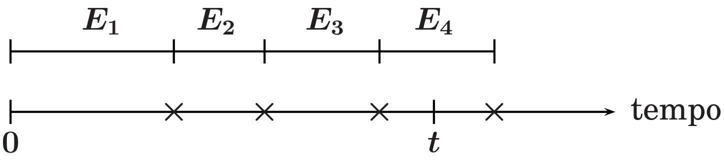
Em uma execução, usando-se a semente 1234 para a RAND2, em que foram gerados 10.000 números com os parâmetros $\alpha = 4$ e $\beta = 1$ da Weibull, os seguintes resultados foram obtidos:
Primeiros 5 números gerados: 6.033, 8.615, 11.382, 5.210 e 0.457.
A média dos 10.000 números gerados foi igual a 4.0048. A média teórica é igual a 4.
O desvio padrão obtido foi igual a 3.9575. O desvio padrão teórico é igual a 4.

4.2 Simulação Direta

O método da Transformação Inversa só pode ser usado se uma expressão analítica puder ser obtida para a função de distribuição acumulada e ela possa ser resolvida explicitamente para x. Existem muitas situações onde isto não é possível, como para a distribuição normal, por exemplo. Uma técnica alternativa para estes casos é o uso da simulação direta do processo sob consideração.

4.2.1 A distribuição de Poisson

A distribuição de Poisson está intimamente relacionada com a distribuição exponencial e é usada em muitos problemas de simulação que envolvem chegadas e partidas. Em particular, se o tempo entre sucessivas chegadas (ou partidas) é exponencialmente distribuído, então o número de eventos que ocorrem em um intervalo de tempo finito t será distribuído de acordo com uma distribuição de Poisson.
Suponha que as marcações ($\times$) na linha do tempo mostrada abaixo, sejam os instantes da chegada de clientes em um posto bancário:



Seja E_i uma variável aleatória exponencialmente distribuída com média $1/\lambda t$, ou seja o intervalo entre as chegadas.

Se fazemos $S_k = \sum_{i=1}^{k} E_i$, então para o intervalo t, temos:
$S_k \leq t < S_{k+1}$, onde k é a variável Poisson, com média λt, ou seja o número de chegadas no intervalo t ($k = 3$ no gráfico acima).
Já vimos (página 57) que podemos gerar variáveis exponenciais (E_i) através da relação:

$$E_i = -\frac{1}{\lambda} \ln U_i$$

Logo a variável "poisson" é o maior valor de k que garante que:

$$t \geq \sum_{i=1}^{k} E_i$$

ou

$$t \geq \sum_{i=1}^{k} -\frac{1}{\lambda} \ln U_i$$

que pode ser escrita como:

$$-\lambda t \geq \sum_{i=1}^{k} \ln U_i$$

ou

$$-\lambda t \geq \ln \prod_{i=1}^{k} U_i$$

Fazendo $t = 1$ e exponenciando ambos os lados chegamos a:

$$\prod_{i=1}^{k} U_i \leq e^{-\lambda}$$

Assim sendo, a variável "poisson" é o maior valor de k que garante a relação acima ou $(k - 1)$ quando na execução do produtório, acontece:

$$\prod_{i=1}^{k} U_i > e^{-\lambda}$$

Exemplo: Gerar 5 variáveis aleatórias governadas pela distribuição de Poisson com média $(\lambda) = 1.5$. Faça os cálculos usando o seguinte conjunto de números aleatórios uniformemente distribuídos: 0.35, 0.97, 0.22, 0.15, 0.60, 0.43, 0.79, 0.52, 0.81, 0.65, 0.20, 0.57, 0.10.
Como $\lambda = 1.5$ temos $e^{-\lambda} = 0.223$. De maneira a obter a primeira variável aleatória, 3 números aleatórios uniformemente distribuídos são necessários como podemos ver a seguir:
$U_1 \times U_2 = (0.35)(0.97) = 0.340 > 0.223$
$U_1 \times U_2 \times U_3 = (0.35)(0.97)(0.22) = 0.075 < 0.223$
Logo $X_1 = (3 - 1) = 2$.
De forma semelhante obtemos: $X_2 = 0$ pois $0.15 < 0.223$
$X_3 = 2$ pois $(0.60)(0.43)(0.79) = 0.204 < 0.223$
$X_4 = 3$ pois $(0.52)(0.81)(0.65)(0.20) = 0.055 < 0.223$
$X_5 = 1$ pois $(0.57)(0.10) = 0.057 < 0.223$
O programa a seguir, escrito em Pascal, implementa a geração de números aletórios que seguem a distribuição de Poisson.
O programa permite que se informe quantos números queremos gerar e o valor de λ.
Como saída ele imprime os 5 primeiros números gerados e a média de todos os números gerados.
O programa usa a RAND2 como o gerador básico de números aleatórios.

```
{POISSON}
Var
  N, I ,NUMIMP, J : Integer;
  SOMA: Double;
  LAMBDA,MEDIA,F: Real;
{$I RAND2.PAS}
{**************************************************}
Function POISSON (F: Real): Integer;
Var
  ALEAT : Double;
  VMULT : Real;
  NUMPOISSON : Integer;
Begin
    NUMPOISSON := 0;
    VMULT := 1.0;
    While (VMULT > F) do
      Begin
        ALEAT := RAND2;
        VMULT := VMULT * ALEAT;
        NUMPOISSON := NUMPOISSON + 1;
      End;
    POISSON := NUMPOISSON - 1;
End;
{**************************************************}
Begin
    Writeln('Qual a semente ? (1 - 2147483646) ');
    Readln(semente);
    Writeln('Quantos Numeros ?');
    Readln(N);
    Writeln('Qual o valor de Lambda ?');
    Readln(LAMBDA);
    F := EXP(-LAMBDA);
    SOMA := 0.0;
    NUMIMP := 0;
    For I := 1 to N do
      Begin
        J := POISSON(F);
        SOMA := SOMA + J;
        NUMIMP := NUMIMP + 1;
        If NUMIMP < 6 then WRITELN(J);
      End;
    Writeln('MEDIA = ',SOMA / N:10:5);
End.
```

Em uma execução, usando-se a semente 9999 para a RAND2, em que foram gerados 10.000 números com $\lambda = 1.5$, os seguintes resultados foram obtidos:
Primeiros 5 números gerados: 0, 1, 1, 2 e 1.
A média (λ) dos 10.000 números gerados foi igual a 1.5053.

4.2.2 A Distribuição Gamma

Vamos ver o uso da Simulação Direta para a Distribuição Gamma, cuja densidade probabilística é dada por:

$$f(x) = \frac{\alpha^{\beta} x^{(\beta-1)} e^{-\alpha x}}{(\beta-1)!}$$

onde α é uma constante positiva e β é uma constante inteira positiva. Pode ser mostrado que a média e a variância para a distribuição são:

$$\mu = \frac{\beta}{\alpha}$$
$$\alpha^2 = \frac{\beta}{\alpha^2} = \frac{\mu}{\alpha}$$

Pode-se mostrar ainda que a variável x pode ser interpretada como a soma de β variáveis aleatórias exponencialmente distribuídas, cada uma tendo um valor esperado de $\frac{1}{\alpha}$. Assim,

$$x = x_1 + x_2 + \ldots + x_{\beta}$$

onde

$$f(x_i) = \alpha e^{-\alpha x_i}$$

A função de densidade probabilística para a distribuição gamma não pode ser integrada analíticamente e consequentemente o método da transformação inversa não pode ser usado. Nós podemos, entretanto, simular o processo gamma diretamente, somando β variáveis aleatórias exponencialmente distribuídas. Assim podemos escrever:

$$X = -\left(\frac{1}{\alpha}\right) \sum_{i=1}^{\beta} \ln U_i$$

onde U_i é uma variável aleatória uniformemente distribuída em $[0, 1]$.
A expressão pode ser escrita de uma forma mais conveniente como:

$$X = -\left(\frac{1}{\alpha}\right) \ln \prod_{i=1}^{\beta} U_i$$

desde que o logarítimo de um produto é a soma dos logarítimos dos fatores individuais.

Exemplo: Gerar 5 variáveis governadas pela distribuição gamma com $\alpha = 1$ e

$\beta = 2$. Faça os cálculos usando o seguinte conjunto de variáveis aleatórias uniformemente distribuídas em $[0,1]$: 0.35, 0.97, 0.22, 0.15, 0.60, 0.43, 0.79, 0.52, 0.81, 0.65.

$\boldsymbol{X_1 = -(1/1)\ln\left[(0.35)(0.97)\right] = 1.08}$
$\boldsymbol{X_2 = -(1/1)\ln\left[(0.22)(0.15)\right] = 3.41}$
$\boldsymbol{X_3 = -(1/1)\ln\left[(0.60)(0.43)\right] = 1.35}$
$\boldsymbol{X_4 = -(1/1)\ln\left[(0.79)(0.52)\right] = 0.89}$
$\boldsymbol{X_5 = -(1/1)\ln\left[(0.81)(0.65)\right] = 0.64}$

O programa a seguir, escrito em Pascal, implementa a geração de números aleatórios que seguem a distribuição gamma.
O programa permite que se informe quantos números queremos gerar, α e β.
Como saída ele imprime os 5 primeiros números gerados e a média de todos os números gerados.
O programa usa a RAND2 como o gerador básico de números aleatórios.

```
{GAMMA}
Var
   N, I ,NUMIMP,BETA: Integer;
   ALEAT,SOMA: Double;
   ALFA,MEDIA: Real;
{$I RAND2.PAS}
{*************************************************}
Function GAMMA (ALFA: Real; BETA :Integer): Double;
Var
  NUMGAMMA: Double;
   J: Integer;
Begin
     NUMGAMMA := 1.0;
     For J := 1 to BETA do
        Begin
          NUMGAMMA := NUMGAMMA * RAND2;
        End;
        GAMMA := (-1.0 / ALFA) * LN(NUMGAMMA);
End;
{*************************************************}
Begin
     Writeln('Qual a semente ? (1 - 2147483646) ');
     Readln(semente);
     Writeln('Quantos Numeros ?');
     Readln(N);
     Writeln('Qual o valor de Alfa ?');
     Readln(ALFA);
     Writeln('Qual o valor de Beta ?');
     Readln(BETA);
     SOMA := 0.0;
     NUMIMP := 0;
     For I := 1 to N do
        Begin
          ALEAT := GAMMA(ALFA,BETA);
          SOMA := SOMA + ALEAT;
          NUMIMP := NUMIMP + 1;
           If NUMIMP < 6 then Writeln(ALEAT:10:6);
        End;
    Writeln('MEDIA = ',SOMA / N:10:6);
End.
```

Em uma execução, usando-se a semente 9999 para a RAND2, em que foram gerados 10.000 números com $\alpha = 1$ e $\beta = 2$, ou seja com média teórica igual a $\mu = \frac{\beta}{\alpha} = \frac{2}{1} = 2$, os seguintes resultados foram obtidos:
Primeiros 5 números gerados: 3.637, 2.884, 1.318, 2.106 e 2.348
A média dos 10.000 números gerados foi igual a 1.9981.

4.2.3 A distribuição Normal

Muitos tipos de eventos aleatórios são governados pela distribuição Normal. Esta distribuição é caracterizada por uma densidade probabilística dada por:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2}$$

onde μ é a média e σ é o desvio padrão. A função de densidade normal não pode ser integrada analíticamente e desta forma não podemos usar o método da transformação inversa. Podemos, entretanto, uma vez mais, gerar a variável aleatória desejada por simulação direta.

Para fazer isto considere o caso especial onde $\sigma = 1$ e $Z = \frac{(x-\mu)}{\sigma}$.
Temos então:

$$f(Z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{Z^2}{2}}$$

Esta é a função de densidade probabilística para a distribuição normal padronizada (standard).
Pelo teorema do limite central sabemos que a soma de N variáveis aleatórias uniformemente distribuídas em $[0, 1]$ segue uma distribuição Normal com $\mu = \frac{N}{2}$ e $\sigma = \sqrt{\frac{N}{12}}$.
Podemos escrever:

$$Z = \frac{\sum_{i=1}^{N} U_i - \frac{N}{2}}{\sqrt{\frac{N}{12}}}$$

Como esta consideração é válida para $N > 10$, podemos fazer $N = 12$ para facilitar o procedimento computacional, obtendo então:

$$Z = \sum_{i=1}^{12} U_i - 6$$

Temos agora um procedimento simples para gerar uma variável aleatória normalmente padronizada. Simplesmente somamos 12 números aleatórios uniformemente distribuídos em $[0, 1]$ e então subtraímos 6, obtendo um valor para Z.

Se desejarmos gerar uma variável normal com média μ e desvio padrão σ, geramos primeiro Z e então calculamos a variável aleatória desejada X usando: $X = \mu + \sigma Z$.

Exemplo: Gerar uma variável aleatória que siga a distribuição normal com média 5 e desvio padrão 2. Use o seguinte conjunto de variáveis aleatórias uniformemente distribuídas em $[0, 1]$: 0.35, 0.97, 0.22, 0.15, 0.60, 0.43, 0.79, 0.52, 0.81, 0.65, 0.20, 0.57.

A soma dos 12 números dá:

$$\sum_{i=1}^{12} U_i = 0.35 + 0.97 + \ldots + 0.57 = 6.26$$

Calculamos então o valor de $Z = (6.26 - 6) = 0.26$.
A variável aleatória normal pode então ser obtida por:
$X = 5 + (2)(0.26) = 5.52$
O programa a seguir, escrito em Pascal, implementa a geração de números aleatórios que seguem a distribuição Normal.
O programa permite que se informe quantos números queremos gerar, a média μ e o desvio padrão σ.
Como saída ele imprime os 6 primeiros números gerados, a média de todos os números gerados e o desvio padrão.
O programa usa a RAND2 como o gerador básico de números aleatórios.

```
{NORMAL I}
Var
  N, I ,NUMIMP, J : Integer;
  SOMA, Z ,DESVIO: Double;
  MU,DELTA: Real;
{$I RAND2.PAS}
{*************************************************}
Function NORMAL(MU,DELTA: Real): Double;
Var
  SOMA12 : Double;
Begin
    SOMA12 := 0.0;
    For J := 1 to 12 do
      Begin
        SOMA12 := SOMA12 + RAND2;
      End;
    NORMAL := MU + DELTA * (SOMA12 - 6.0);
End;
{*************************************************}
Begin
    Writeln('Qual a semente ? (1 - 2147483646) ');
    Readln(semente);
    Writeln('Quantos Numeros ?');
    Readln(N);
    Writeln('Qual a media ?');
    Readln(MU);
    Writeln('Qual o desvio padrao ?');
    Readln(DELTA);
    SOMA := 0.0;
    NUMIMP := 0;
```

```
    DESVIO := 0.0;
    For I := 1 to N do
      Begin
        Z := NORMAL(MU,DELTA);
        SOMA := SOMA + Z;
        DESVIO := DESVIO + (Z * Z);
        NUMIMP := NUMIMP + 1;
        If NUMIMP < 6 then Writeln(Z:10:6);
      End;
   Writeln('MEDIA = ',SOMA / N:10:6);
   DESVIO := SQRT((DESVIO / N) - (SOMA /N) * (SOMA / N));
   Writeln('DESVIO PADRAO = ',DESVIO:10:6);
END.
```

Em uma execução, usando-se a semente 9999 para a RAND2, em que foram gerados 10.000 números com $\mu = 5$ e $\sigma = 2$, os seguintes resultados foram obtidos:
Primeiros 5 números gerados: 2.085, 2.899, 0.675, 6.587 e 0.518.
A média (μ) dos 10.000 números gerados foi igual a 5.
O desvio padrão (σ) foi igual a 2.008.

Um método alternativo para gerar variáveis aleatórias normalmente distribuídas é usar uma das seguintes expressões:

$$Z = \sqrt{(-2\ln U_1)}\sin(2\pi U_2)$$

ou

$$Z = \sqrt{(-2\ln U_1)}\cos(2\pi U_2)$$

Ambas as expressões geram variáveis aleatórias normais padronizadas.
Observe que o método anterior necessita de 12 valores de U_i para cada valor de Z enquanto que este último só necessita de 2. Assim aparenta ser mais eficiente do ponto de vista computacional mas o cálculo de logarítimo, raiz quadrada e seno (ou coseno) é muito mais demorado que uma soma. Na verdade os 2 métodos se equivalem em termos de tempo computacional.

Exemplo: Gere 2 números aleatórios que sigam uma distribuição normal com média 5 e desvio padrão 2. Use o seguinte conjunto de números aleatórios uniformemente distribuídos em $[0, 1]$: 0.35, 0.97, 0.22, 0.15.
Temos então:
$Z_1 = \sqrt{(-2)\ln(0.35)}\sin[(2\pi)(0.97)] = -0.27$
$X_1 = 5 + (2)(-0.27) = 4.46$
$Z_1 = \sqrt{(-2)\ln(0.22)}\sin[(2\pi)(0.15)] = 1.41$
$X_2 = 5 + (2)(1.41) = 7.82$
O programa a seguir, escrito em Pascal, implementa a geração de números aleatórios que seguem a distribuição Normal utilizando a fórmula do seno.
O programa permite que se informe quantos números queremos gerar, a média μ e o desvio padrão σ.
Como saída ele imprime os 6 primeiros números gerados, a média de todos os números gerados e o desvio padrão.
O programa usa a RAND2 como o gerador básico de números aleatórios.

```
{NORMAL II}
Var
  N,I,NUMIMP: Integer;
  ALEAT,SOMA,Z,DESVIO: Double;
  MU,DELTA: Real;
{$I RAND2.PAS}
{************************************************}
Function NORMAL(MU,DELTA: Real): Double;
Begin
  ALEAT := SQRT((-2.0 * LN(RAND2))) * SIN(2.0 * PI * RAND2);
  NORMAL := MU + DELTA * ALEAT;
End;
{************************************************}
Begin
    Writeln('Qual a semente ? (1 - 2147483646) ');
    Readln(semente);
    Writeln('Quantos Numeros ?');
    Readln(N);
    Writeln('Qual a media ?');
    Readln(MU);
    Writeln('Qual o desvio padrao ?');
    Readln(DELTA);
    SOMA := 0.0;
    NUMIMP := 0;
    DESVIO := 0.0;
    For I := 1 to N do
      Begin
        Z := NORMAL(MU,DELTA);
        SOMA := SOMA + Z;
        DESVIO := DESVIO + (Z * Z);
        NUMIMP := NUMIMP + 1;
        If NUMIMP < 6 then Writeln(Z:10:6);
      End;
   Writeln('MEDIA = ',SOMA / N:10:6);
   DESVIO := SQRT((DESVIO / N) - (SOMA /N) * (SOMA / N));
   Writeln('DESVIO PADRAO = ',DESVIO:10:6);
End.
```

Em uma execução, usando-se a semente 9999 para a RAND2, em que foram gerados 10.000 números com $\mu = 5$ e $\sigma = 2$, os seguintes resultados foram obtidos:
Primeiros 5 números gerados: 3.882, 5.281, 2.879, 7.491 e 7.621.
A média (μ) dos 10.000 números gerados foi igual a 4.983.
O desvio padrão (σ) foi igual a 2.018.

4.3 O Método da Rejeição

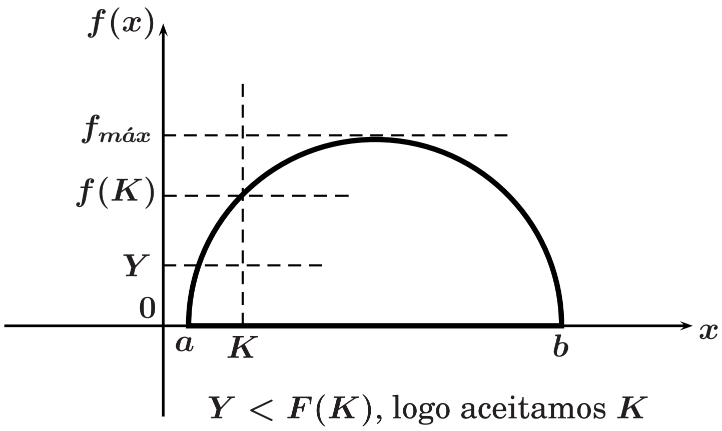
O Método da Rejeição é um procedimento geral para gerar variáveis aleatórias para qualquer distribuição cuja densidade probabilística $f(x)$ é contínua e limitada dentro de uma região finita, isto é, necessitamos que $0 \leq f(x) \leq f_{max}$ dentro do intervalo $a \leq x \leq b$.
De maneira a se obter a variável aleatória X, deve-se proceder da seguinte forma:

1. Gerar um par (U_1, U_2) de números aleatórios uniformemente distribuídos em (0,1).

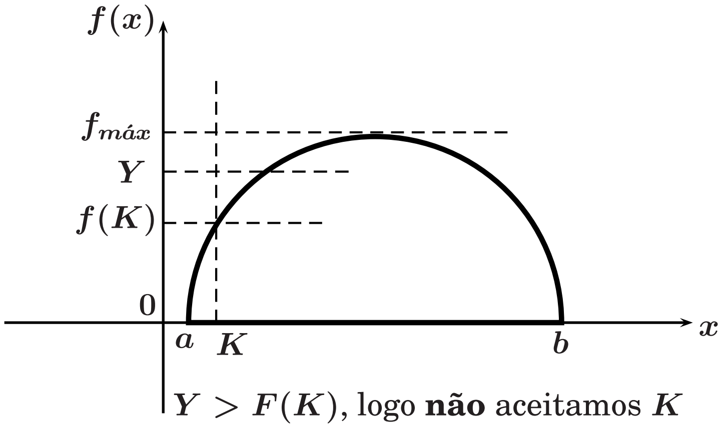
2. Obter uma variável aleatória, K, dentro do intervalo $a \leq K \leq b$, usando a relação $K = a + (b - a) \times U_1$.

3. Avaliar a densidade probabilística no ponto K, isto é, determinar $f(K)$.

4. Obter uma variável aleatória, Y, uniforme dentro do intervalo $0 \leq Y \leq f_{max}$, usando a relação $Y = f_{max} \times U_2$. Os pontos Y e K representam as coordenadas de algum ponto no espaço como ilustrado nas figuras a seguir:



$Y < F(K)$, logo aceitamos K



$Y > F(K)$, logo **não** aceitamos K

5. Comparar Y com $f(K)$.
   Se Y não é maior que $f(K)$ então o ponto (Y, K) cairá em cima ou abaixo da curva de densidade probabilística como indicado na primeira figura acima. Neste caso nós aceitamos K como a variável aleatória desejada, ou seja, fazemos $X = K$.
   Se Y é maior que $f(K)$, rejeitamos o ponto.

6. As etapas de 1 a 5 são repetidas sucessivamente até ser encontrado um ponto que satisfaça a condição.

Embora o método da rejeição possa ser usado com muitas distribuições diferentes, ele é ineficiente por causa das diversas tentativas que se tem que fazer para se obter uma variável aleatória desejada. Por esta razão só deve ser usado se não existir outro método.

4.3.1 A Distribuição Beta

Para ilustrar o uso do método da rejeição, vamos considerar a distribuição Beta. Esta distribuição tem a densidade probabilística dada por:

$$f(x) = \frac{(\beta_1 + \beta_2 - 1)!x^{(\beta_1 - 1)}(1 - x)^{(\beta_2 - 1)}}{(\beta_1 - 1)!(\beta_2 - 1)!}$$

onde β_1 e β_2 são inteiros positivos e $0 \leq x \leq 1$.
Pode ser mostrado que a média e a variância para esta distribuição são:

$$\mu = \frac{\beta_1}{(\beta_1 + \beta_2)}$$

$$\sigma^2 = \frac{\mu\beta_2}{(\beta_1 + \beta_2) + (\beta_1 + \beta_2 + 1)}$$

Como $0 \leq x \leq 1$ para a distribuição Beta, temos, no método da rejeição, $a = 0$ e $b = 1$.
Exemplo: O modo mais fácil de gerar uma variável aleatória Beta é usar simulação direta. Vamos no entanto, como exemplo, usar o método da rejeição. Em particular vamos gerar várias variáveis beta com $\beta_1 = 2$ e $\beta_2 = 3$, baseado na seguinte sequência de números aleatórios uniformemente distribuídos em (0,1): 0.35, 0.97, 0.22, 0.15, 0.60, 0.43, 0.79, 0.52, 0.81, 0.65, 0.20, 0.57.
$K = a + (b - a) \times U_1 = 0 + (1 - 0) \times U_1$
$K = U_1$
A função de densidade probabilística pode ser escrita como:
$f(x) = \dfrac{(2 + 3 - 1)!x^{(2-1)}(1 - x)^{(3-1)}}{(2 - 1)!(3 - 1)!}$
$f(x) = 12x(1 - x)^2$

Esta função tem o seu valor máximo em $x = 1/3$.
Temos então:
$f_{max} = 12\left(\frac{1}{3}\right)\left(\frac{2}{3}\right)^2 = \frac{16}{9} = 1.78$
Como Y é igual $f_{max} \times U_2$, temos:
$Y = 1.78 \times U_2$
Os resultados obtidos com a aplicação do método estão mostrados a seguir:

U_1	U_2	K	$f(K)$	Y	$Y \leq f(K)$?	X
0.35	0.97	0.35	1.77	1.73	SIM	0.35
0.22	0.15	0.22	1.61	0.27	SIM	0.22
0.60	0.43	0.60	1.15	0.77	SIM	0.60
0.79	0.52	0.79	0.42	0.93	NÃO	
0.81	0.65	0.81	0.35	1.16	NÃO	
0.20	0.57	0.20	1.54	1.01	SIM	0.20

Assim com 12 números aleatórios em $[0, 1]$, nós geramos 4 variáveis beta cujos valores são 0.35, 0.22, 0.60 e 0.20.
O programa a seguir, escrito em Pascal, implementa a geração de números aleatórios que seguem a distribuição Beta utilizando o método da rejeição.
O programa permite que se informe quantos números queremos gerar e o valor f_{max}.
Como saída ele imprime os 5 primeiros números gerados, a média de todos os números gerados e o desvio padrão.
Ele imprime também quantos números aleatórios em $[0, 1]$ foram necessários.
O programa usa a RAND2 como o gerador básico de números aleatórios.

```
{BETA}
Const
A = 0;
B = 1;
Var
   N, I ,NUMIMP:  Integer;
   J:  Longint;
   U:  Double;
   MEDIA,FMAX,K,Y,SOMA,DESVIO,NUMBETA:  Real;
{$I RAND2.PAS}
Begin
     Writeln('Qual a semente ? (1 - 2147483646) ');
     Readln(semente);
     Writeln('Quantos Numeros ?');
     Readln(N);
     Writeln('Qual o valor de Fmax ?');
     Readln(FMAX);
     SOMA := 0.0;
     NUMIMP := 0;
     DESVIO := 0.0;
     I := 0;
     J := 0;
     While I < N do
```

```
      Begin
        U := RAND2;
        J := J +1;
        K := A + (B - A) * U;
        U := RAND2;
        J := J + 1;
        Y := FMAX * U;
        If Y < (12 * K * (1 - K) * (1 - K)) then
           Begin
             I := I + 1;
             NUMBETA := K;
             SOMA := SOMA + NUMBETA;
             DESVIO := DESVIO + NUMBETA * NUMBETA;
             NUMIMP := NUMIMP + 1;
             If NUMIMP < 6 then Writeln(NUMBETA:10:6);
           End;
      End;
     Writeln('Usados ',J,' numeros aleatorios');
     Writeln('MEDIA = ',SOMA / N:10:6);
     SOMA := SOMA / N;
     DESVIO := SQRT((DESVIO / N) - (SOMA * SOMA));
     Writeln('Desvio Padrao = ',DESVIO:10:6);
End.
```

Em uma execução, usando-se a semente 9999 para a RAND2, em que foram gerados 10.000 números, os seguintes resultados foram obtidos:
Primeiros 5 números gerados: 0.114, 0.304, 0.372, 0.273 e 0.187
A média (μ) dos 10.000 números gerados foi igual a 0.4018. A média teórica é igual a:

$$\mu = \frac{\beta_1}{\beta_1 + \beta_2} = \frac{2}{2+3} = 0.4$$

O desvio padrão (σ) foi igual a 0.2007. O desvio padrão teórico é igual a:

$$\sigma = \sqrt{\frac{\mu\beta_2}{(\beta_1+\beta_2)(\beta_1+\beta_2+1)}} = \sqrt{\frac{0.4 \times 3}{(2+3)(2+3+1)}} = 0.2$$

Para se gerar os 10.000 números beta, foram necessários 35.328 números aleatórios gerados pela RAND2, o que mostra a ineficiência do método.

4.4 Outras funções de distribuição

Vimos algumas das funções de distribuição mais importantes. Devemos assinalar que existem métodos e formas de gerar variáveis aleatórias para praticamente qualquer distribuição conhecida e estudada.

Capítulo 5

Modelos para simular filas de espera e o uso do ARENA

Neste capítulo veremos, inicialmente, 2 programas que permitem a simulação de alguns modelos de filas de espera. A seguir, veremos uma introdução ao **ARENA**, que é um programa para se construir e simular modelos no ambiente Windows.

5.1 Simulação de um pequeno posto bancário

Veremos uma aplicação construída para simular o funcionamento de um pequeno posto bancário com uma único caixa e uma única fila.
Dados obtidos por amostragem indicam que o intervalo entre chegadas de clientes segue uma Distribuição Exponencial e a duração do serviço prestado, pelo caixa, também segue uma Distribuição Exponencial.
O programa a seguir, escrito em Pascal, implementa a simulação desejada e utiliza a RAND2 como o gerador de números aleatórios uniformemente distribuídos.
Neste tipo de simulação, em que temos 2 eventos aleatórios (intervalo entre chegadas e duração do serviço), devemos utilizar 2 sementes diferentes, uma para cada tipo de evento.

```
{Fila com 1 servidor:  Intervalo entre chegadas: Exponencial
Ex. Posto bancario      Duracao do servico: Exponencial
    com 1 caixa
Usa a RAND2 como gerador de nos. aleatorios [0,1].
***********************************************************************
Variaveis principais usadas no programa
CLIENTES_NA_FILA - Numero de clientes esperando na fila
CLIENTES_RECEBENDO_SERVICO - Clientes sendo servidos (0 ou 1)
CHEGADA(I) - Instante da chegada do (iesimo -1) cliente
             a ser atendido, assim CHEGADA(2) e' o instante
             de chegada do primeiro na fila de espera e
             CHEGADA(1) e' o instante da chegada
             do cliente sendo atendido
INSTANTE_PROXIM0_EVENTO(I) - Instante da ocorrencia do proximo evento
                             [1 (chegada) ou 2 (saida)]
TIPO_PROXIMO_EVENTO - Tipo de proximo evento (1 ou 2)
DURACAO_SERVICO - Duracao do servico do ultimo cliente a receber servico
RELOGIO - Relogio de controle da simulacao
NUMERO_EVENTOS - Numero de eventos (=2 ou seja, chegadas e servicos prestados)
```

```
INSTANTE_ULTIMO_EVENTO - Instante da ocorrencia do ultimo evento
                         usada para atualizar TEMPO_SERVIDOR_OCUPADO
TEMPO_SERVIDOR_OCUPADO - Tempo total de ocupacao da caixa
TEMPO_NO_SISTEMA - Tempo que um cliente fica no sistema
SOMA_TEMPOS_NO_SISTEMA - Soma tempos no sistema de todos os clientes
NO_SERVICOS_PRESTADOS - Numero de servicos prestados, ou seja, clientes atendidos
MAIS_DE_4 - Numero de clientes que ficaram mais de 4 unidades de tempo na fila
MAXIMO_NA_FILA - Maior numero de clientes na fila durante toda a simulacao
INSTANTE_ULTIMO_EVENTO - Instante da ocorrencia do ultimo evento
NUMERO_DE_CLIENTES - numero de clientes tratados pela simulacao
RHO - Taxa de ocupacao do servidor
PERC_MAIS_4 - Percentual de clientes que ficam mais de 4 unidades de
              tempo na fila
TEMPO_TOTAL_NA_FILA - Soma dos tempos gasto pelos clientes na fila
UNIDADE_DE_TEMPO - Unidade de medicao do tempo
*********************************************************************}
Uses crt;
Label
VOLTA;
Var
G1,G2, U: Double;
MEDIAEXP,ALFA,MEDIAEXP1,ALFA1: Real;
{*********************************************************************}
MAXIMO_NA_FILA,TIPO_PROXIMO_EVENTO,CLIENTES_RECEBENDO_SERVICO: Integer;
NUMERO_DE_CLIENTES,NO_SERVICOS_PRESTADOS,MAIS_DE_4: Longint;
NUMERO_EVENTOS, I: Integer;
CLIENTES_NA_FILA: Longint;
RELOGIO,TEMPO_NO_SISTEMA,TEMPO_SERVIDOR_OCUPADO,FMIN,DURACAO_SERVICO: Real;
INSTANTE_ULTIMO_EVENTO,SOMA_TEMPOS_NO_SISTEMA,RHO,PERC_MAIS_4: Real;
TEMPO_TOTAL_NA_FILA: Real;
UNIDADE_DE_TEMPO: String[9];
INSTANTE_PROXIMO_EVENTO: Array[1..2] of Real;
CHEGADA: Array[1..100] of Real;
{*********************************************************************}
{$I RAND2.PAS}
{/////////////////////////////////////////////////////////////////////
Funcao para a geracao de nos. aleatorios exponencialmente distribuidos
*********************************************************************}
Function EXPONENCIAL(ALFA: Real; U: Double): Real;
 Begin
   EXPONENCIAL:= -(1/ALFA) * LN(U);
 End;
{/////////////////////////////////////////////////////////////////////
Inicializacao
*********************************************************************}
Procedure INICIALIZACAO;
      Begin
      NUMERO_EVENTOS:=2;
      RELOGIO:=0.0;
      CLIENTES_NA_FILA:=0;
      CLIENTES_RECEBENDO_SERVICO:=0;
      TEMPO_SERVIDOR_OCUPADO:=0.0;
      MAXIMO_NA_FILA:=0;
```

```
    SOMA_TEMPOS_NO_SISTEMA:=0;
    MAIS_DE_4:=0;
    NO_SERVICOS_PRESTADOS:=0;
    TEMPO_TOTAL_NA_FILA:= 0.0;
{************************************************************************
Gera a primeira chegada
Faz INSTANTE_PROXIMO_EVENTO[2] igual a infinito para indicar que uma
partida e' impossivel com o sistema vazio
************************************************************************}
    semente:= G1;
    U:= RAND2;
    G1:= semente;
    INSTANTE_PROXIMO_EVENTO[1]:= RELOGIO + EXPONENCIAL(ALFA,U);
    INSTANTE_PROXIMO_EVENTO[2]:= 1.0E30;
End;
{///////////////////////////////////////////////////////////////////////
Rotina de avanco do tempo para determinar proximo evento
e avancar relogio para o instante do proximo evento
************************************************************************}
Procedure AVANCA;
    Begin
    FMIN:=1.0E20;
    TIPO_PROXIMO_EVENTO:=0;
    For I:= 1 to NUMERO_EVENTOS do
      Begin
        If INSTANTE_PROXIMO_EVENTO[I] < FMIN Then
          Begin
            FMIN := INSTANTE_PROXIMO_EVENTO[I];
            TIPO_PROXIMO_EVENTO:=I;
          End;
      End;
    If TIPO_PROXIMO_EVENTO = 0 Then
      Begin
        Writeln('LISTA DE EVENTOS FUTUROS VAZIA - ERRO');
        HALT;
      End;
    RELOGIO:=INSTANTE_PROXIMO_EVENTO[TIPO_PROXIMO_EVENTO];
End;
{///////////////////////////////////////////////////////////////////////
Tratamento de uma chegada
************************************************************************}
Procedure CHEGADAS;
{***********************************************************************
Determina se o servidor esta' ocupado. Menor que 1 esta' livre.
************************************************************************}
Begin
    If CLIENTES_RECEBENDO_SERVICO < 1 Then
      Begin
        CLIENTES_RECEBENDO_SERVICO:= 1;
        CHEGADA[1]:=RELOGIO;
{***********************************************************************
Gera uma duracao de servico para a nova chegada e programa a
partida desta nova chegada
```

```
*********************************************************************}
         semente:= G2;
         U:= RAND2;
         G2:= semente;
         DURACAO_SERVICO:= EXPONENCIAL(ALFA1,U);
         INSTANTE_PROXIMO_EVENTO[2]:=RELOGIO + DURACAO_SERVICO;
{*********************************************************************
Atualiza MAXIMO_NA_FILA e INSTANTE_ULTIMO_EVENTO
*********************************************************************}
         INSTANTE_ULTIMO_EVENTO:= RELOGIO;
         If CLIENTES_NA_FILA > MAXIMO_NA_FILA Then
                                        MAXIMO_NA_FILA := CLIENTES_NA_FILA;
      End
{*********************************************************************
O servidor esta' ocupado. Atualiza o estado do sistema e registra
instante da nova chegada
*********************************************************************}
      Else
         Begin
           CLIENTES_NA_FILA:=CLIENTES_NA_FILA + 1;
           I:= CLIENTES_NA_FILA + 1;
           If I > 100 Then
             Begin
               Writeln('Fila maior que 100. Aumente dimensoes');
               HALT;
             End;
           CHEGADA[I]:=RELOGIO;
{*********************************************************************
Atualiza estatisticas cumulativas TEMPO_SERVIDOR_OCUPADO e MAXIMO_NA_FILA
*********************************************************************}
           TEMPO_SERVIDOR_OCUPADO := TEMPO_SERVIDOR_OCUPADO +
                                     (RELOGIO - INSTANTE_ULTIMO_EVENTO);
           INSTANTE_ULTIMO_EVENTO := RELOGIO;
           IF CLIENTES_NA_FILA >= MAXIMO_NA_FILA Then
                                        MAXIMO_NA_FILA := CLIENTES_NA_FILA;
         End;
{*********************************************************************
Gera um intervalo entre chegadas e programa o proximo evento de chegada
*********************************************************************}
     semente:= G1;
     U:= RAND2;
     G1:= semente;
     INSTANTE_PROXIMO_EVENTO[1]:=RELOGIO+EXPONENCIAL(ALFA,U);
End;
{/////////////////////////////////////////////////////////////////////
Rotina de servico
*********************************************************************}
Procedure SERVICO;
{*********************************************************************
Atualiza estatisticas cumulativas TEMPO_SERVIDOR_OCUPADO,
SOMA_TEMPOS_NO_SISTEMA, NO_SERVICOS_PRESTADOS e MAIS_DE_4MIN.
CLIENTES_NA_FILA e' diminuida de modo que MAXIMO_NA_FILA nao muda.
*********************************************************************}
```

```
Begin
    TEMPO_SERVIDOR_OCUPADO:= TEMPO_SERVIDOR_OCUPADO +
                            (RELOGIO - INSTANTE_ULTIMO_EVENTO);
    INSTANTE_ULTIMO_EVENTO := RELOGIO;
    TEMPO_NO_SISTEMA:= RELOGIO - CHEGADA[1];
    SOMA_TEMPOS_NO_SISTEMA:= SOMA_TEMPOS_NO_SISTEMA + TEMPO_NO_SISTEMA;
    NO_SERVICOS_PRESTADOS:= NO_SERVICOS_PRESTADOS + 1;
    If TEMPO_NO_SISTEMA > 4 Then MAIS_DE_4 := MAIS_DE_4 + 1;
{*********************************************************************
Verifica a condicao da fila
*********************************************************************}
    If CLIENTES_NA_FILA < 1 Then
{*********************************************************************
Fila vazia. Servidor desocupado. Proxima saida igual a infinito
*********************************************************************}
      Begin
        CLIENTES_RECEBENDO_SERVICO := 0;
        INSTANTE_PROXIMO_EVENTO[2]:=1.0E30;
      End
{*********************************************************************
Pelo menos 1 cliente na fila. Mover cada cliente 1 posicao
*********************************************************************}
    Else
      Begin
        For I:=1 to CLIENTES_NA_FILA do
          Begin
            CHEGADA[I]:=CHEGADA[I+1];
          End;
{*********************************************************************
Atualiza estado do sistema
*********************************************************************}
        CLIENTES_NA_FILA:= CLIENTES_NA_FILA -1;
        TEMPO_TOTAL_NA_FILA:=TEMPO_TOTAL_NA_FILA + (RELOGIO - CHEGADA[1]);
{*********************************************************************
Gera nova duracao de servico para o cliente chegando para servico
e programa proxima saida
*********************************************************************}
        semente:= G2;
        U:= RAND2;
        G2:= semente;
        DURACAO_SERVICO:= EXPONENCIAL(ALFA1,U);
        INSTANTE_PROXIMO_EVENTO[2]:=RELOGIO + DURACAO_SERVICO;
      End;
End;
{/////////////////////////////////////////////////////////////////////
Impressao dos resultados
*********************************************************************}
Procedure RESULTADOS;
Begin
    RHO:=TEMPO_SERVIDOR_OCUPADO/RELOGIO;
    PERC_MAIS_4:=MAIS_DE_4/NO_SERVICOS_PRESTADOS;
    Writeln('Simulacao de um Sistema de Fila com 1 estacao de servico');
    Writeln('--------------------------------------------------------');
```

```
    Writeln('Intervalo entre chegadas (Exponencial) = ',MEDIAEXP:3:3,
                                          ' ',UNIDADE_DE_TEMPO);
    Writeln('Duracao do servico (Exponencial) = ',MEDIAEXP1:2:3,
                                                 ' ',UNIDADE_DE_TEMPO);
    Writeln('-------------------------------------------------------');
    Writeln('Total de clientes atendidos = ',NO_SERVICOS_PRESTADOS);
    Writeln('Taxa de ocupacao do servidor = RHO = ',RHO:0:3);
    Writeln('Maximo de clientes na fila = ',MAXIMO_NA_FILA);
    Writeln('Percentual de clientes que esperaram mais de 4 ',UNIDADE_DE_TEMPO,
                                          ' = ',PERC_MAIS_4*100:0:2,' %');
    Writeln('Tempo medio gasto por cliente na fila = Wq = ',
     (TEMPO_TOTAL_NA_FILA/NO_SERVICOS_PRESTADOS):3:3,' ',UNIDADE_DE_TEMPO);
    Writeln('Tempo medio gasto por cliente no sistema = W = ',
       (SOMA_TEMPOS_NO_SISTEMA/NO_SERVICOS_PRESTADOS):3:3,' ',UNIDADE_DE_TEMPO);
    Writeln('Tempo total de simulacao = ',RELOGIO:6:3,' ',UNIDADE_DE_TEMPO);
End;
{//////////////////////////////////////////////////////////////////////
ROTINA PRINCIPAL
**********************************************************************
Recebe os parametros de entrada e sementes para a geracao de
numeros aleatorios
**********************************************************************}
Begin
    CLRSCR;
VOLTA:
    Writeln('Qual a semente ? (1 - 2147483646) - Intervalo entre Chegadas ');
    Readln(semente);
    G1:= semente;
    Writeln('Qual a semente ? (1 - 2147483646) - Duracao do Atendimento ');
    Readln(semente);
    G2 := semente;
    IF G1 = G2 Then
       Begin
         Writeln('SEMENTES IGUAIS - ERRO!!');
         GOTO VOLTA;
       End;
    Writeln('Qual a unidade de tempo a ser usada ?');
    Readln(UNIDADE_DE_TEMPO);
    Writeln('Qual o intervalo, em ',UNIDADE_DE_TEMPO,
                             ' ,entre chegadas - EXPONENCIAL ?');
    Readln(MEDIAEXP);
    ALFA:=1.0/MEDIAEXP;
    Writeln('Qual a media, em ',UNIDADE_DE_TEMPO,
                        ',da duracao do servico - EXPONENCIAL ?');
    Readln(MEDIAEXP1);
    ALFA1:= 1.0/MEDIAEXP1;
    Writeln('Quantos clientes na simulacao ?');
    Readln(NUMERO_DE_CLIENTES);
    CLRSCR;
{*******************************************************************}
    INICIALIZACAO;
    While (NO_SERVICOS_PRESTADOS < NUMERO_DE_CLIENTES) do
      Begin
```

```
        AVANCA;
        If TIPO_PROXIMO_EVENTO = 1 Then CHEGADAS
           Else SERVICO;
      End;
      RESULTADOS;
End.
```

A execução deste programa com os seguintes dados de entrada:
Semente para a RAND2, para o intervalo entre chegadas: 4444
Semente para a RAND2, para a duração do serviço: 12345
Intervalo entre chegadas (exponencial): 2 minutos
Duração do Serviço: 1.5 minutos
Número de usuários a serem simulados: 10.000
Apresentou a seguinte tela de saída:

Simulacao de um Sistema de Fila com 1 estacao de servico

Intervalo entre chegadas (Exponencial) = 2.000 minutos
Duracao do servico (Exponencial) = 1.500 minutos

Total de clientes atendidos = 10000
Taxa de ocupacao do servidor = RHO = 0.764
Maximo de clientes na fila = 21
Percentual de clientes que esperaram mais de 4 minutos = 51.89 %
Tempo medio gasto por cliente na fila = Wq = 4.563 minutos
Tempo medio gasto por cliente no sistema = W = 6.064 minutos
Tempo total de simulacao = 19667.265 minutos

Este tipo de fila, tem solução analítica pois se trata do chamado Modelo M/M/1 da Teoria das Filas.
Aplicando as fórmulas teóricas chegamos aos seguintes resultados:
Taxa de ocupação do servidor = ρ (rho) = 0.750
Percentual de clientes que esperaram mais de 4 minutos = 51.34 %
Tempo médio gasto por cliente na fila $= W_q = 4.5$ minutos
Tempo médio gasto por cliente no sistema $= W = 6.0$ minutos
Comparando-se os resultados da simulação com os resultados teóricos, podemos observar que a simulação apresenta resultados totalmente aderentes aos resultados teóricos o que demonstra a potencialidade da técnica.
Obviamente não teria sentido se construir um modelo de simulação para a situação acima onde temos solução analítica. O objetivo foi apenas mostrar o uso das técnicas de simulação.

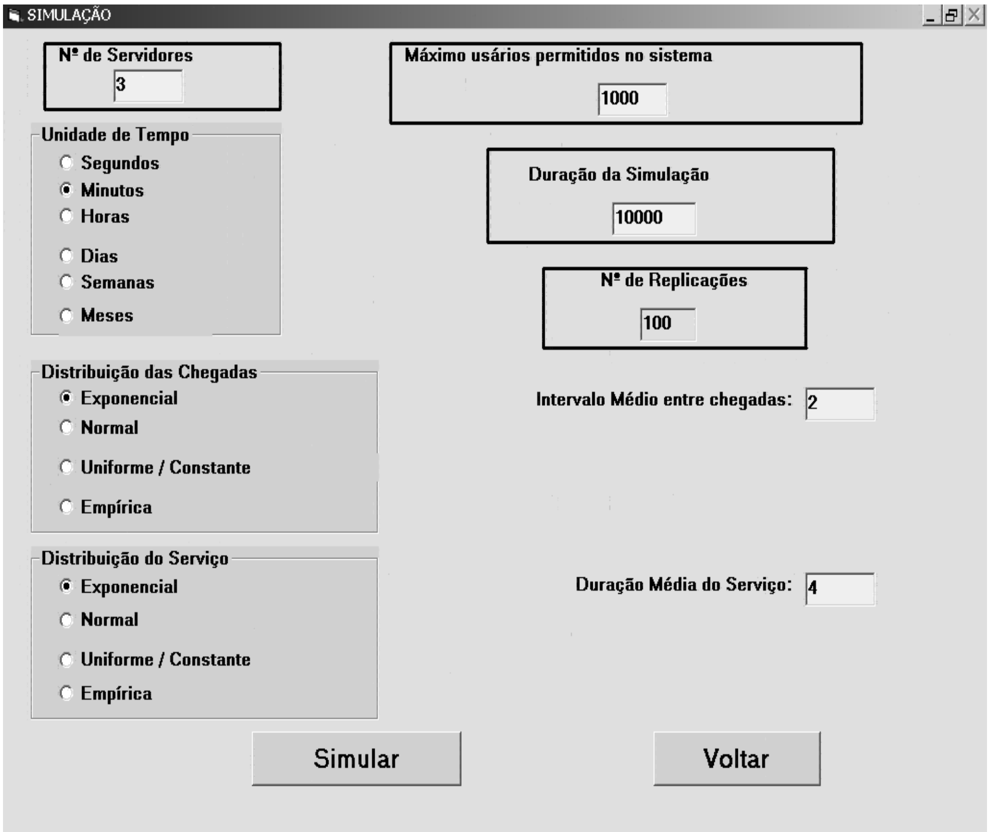
5.2 Um software mais versátil

Veremos a seguir um outro programa (simulacao)[1], um pouco mais versátil do que acabamos de ver. Ele simula um sistema de filas e permite que se escolha o número de estações que prestam serviço, o número máximo de clientes que podem ficar no sistema, a distribuição do intervalo entre chegadas (exponencial, normal, uniforme ou empírica), a distribuição da duração do serviço prestado (exponencial, normal, uniforme ou empírica) e a duração da simulação. Pode-se escolher também a unidade de tempo a ser usada.
Pode-se determinar também o número de replicações (até o máximo de 100), ou seja quantas vezes deseja-se executar a simulação para o mesmo conjunto de dados de entrada.
O programa usa a RAND4 como o gerador de números aleatórios e usa "séries" diferentes, entre 1 e 50, para o intervalo entre chegadas e para a duração do serviço. As sementes e as séries usadas são escolhidas, aleatoriamente, pelo programa. No caso de se executar replicações, séries e sementes diferentes são criadas em cada uma das execuções individuais da replicação.
Os resultados mostrados na saída são, no caso das replicações, as médias obtidas com as diversas execuções.

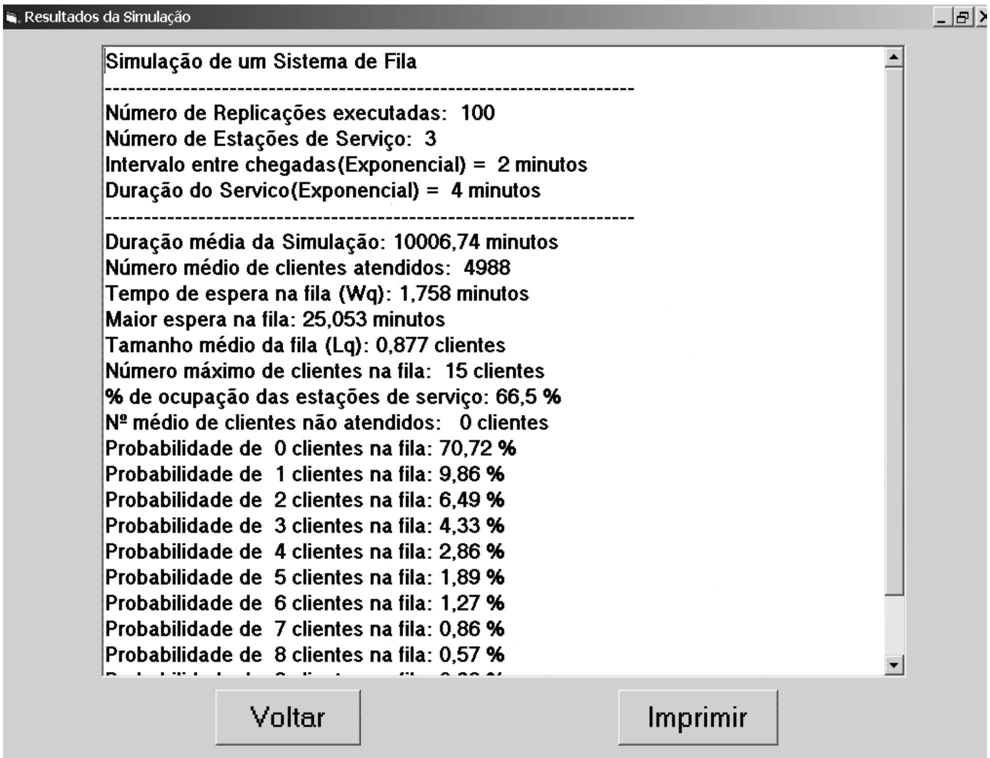
[1]É um dos módulos do programa PO, que pode ser obtido de www.mpsantos.com.br

5.2.1 Alguns exemplos usando o programa "Simulação"

Como exemplo do uso do programa, vamos simular um sistema de filas que tenha solução analítica para compararmos os resultados.
Vamos simular o sistema de filas de uma agência bancária, com 3 caixas de atendimento, em que as chegadas sigam uma distribuição exponencial com intervalo médio entre chegadas igual a 2 minutos.
A distribuição do tempo de atendimento dos caixas também segue uma distribuição exponencial com atendimento médio igual a 4 minutos.
A tela de entrada do programa para execução desta simulação seria a seguinte:



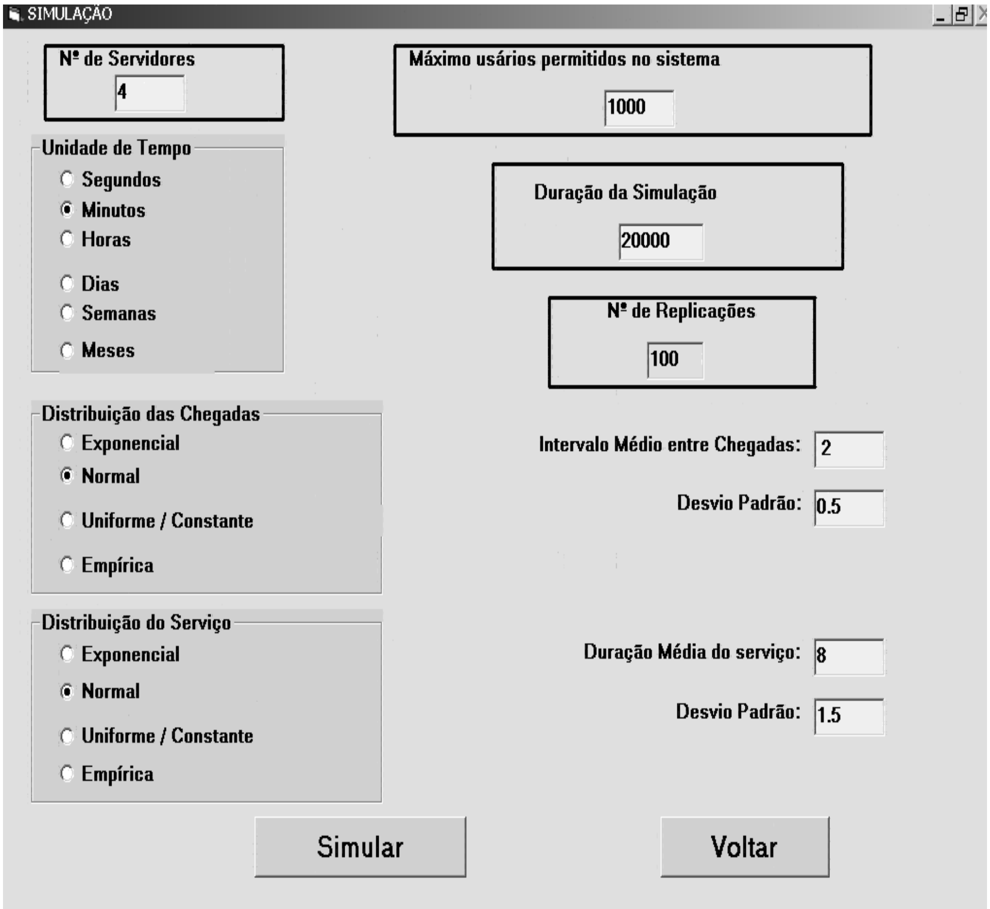
Como não existe limitação para o número de clientes na agência, o campo "Máximo usuários permitido no Sistema" ficou com 1000. A duração da simulação foi de 10.000 minutos e solicitamos que 100 replicações fossem executadas.
O valor em qualquer dos campos pode ser mudado, bastando para isto clicar nele e apagar o valor existente com as teclas **[Delete]** ou **[Backspace]**. Os campos Unidade de tempo, Distribuição das chegadas e Distribuição do Serviço podem ser mudados com o simples click do mouse no novo ítem desejado.
Após se colocar os valores desejados, clica-se em SIMULAR e a Simulação será executada. No nosso exemplo aparecerá a seguinte tela:



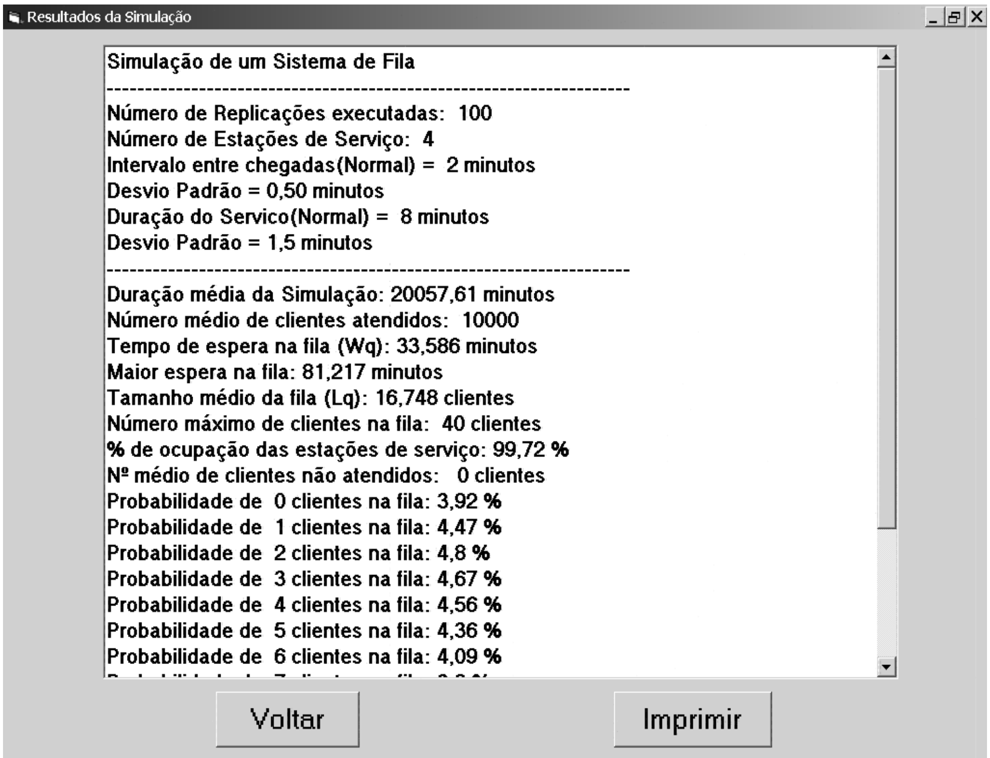
Caso se tenha uma impressora esta tela poderá ser impressa. Clicando-se no botão VOLTAR saímos do programa, voltando-se para o software Po.

Como este modelo simulado é um exemplo do modelo M/M/s, podemos comparar os resultados obtidos na simulação com os resultados teóricos obtidos das fórmulas analíticas daquele modelo de filas:
Tempo médio que um cliente permanece na fila (W_q):
Simulação : 1,758 minutos
Teórico : 1,776 minutos
Número Médio de clientes na fila (L_q):
Simulação : 0,877 clientes
Teórico : 0,888 clientes
Taxa de ocupação das estações de serviço (caixas):
Simulação : 66,5%
Teórico : 66,7%
Podemos observar que a simulação deu resultados praticamente iguais aos teóricos o que demonstra a robustez da técnicas de simulação além de servir para demonstrar que o programa está se comportando de maneira adequada.

Vamos usá-lo agora para simular uma situação de fila em que não existem, pelo menos de forma fácil, fórmulas teóricas.
Vamos supor uma agência bancária, com 4 caixas de atendimento, onde as chegadas chegam seguindo uma distribuição normal com intervalo médio entre chegadas de 2 minutos e desvio padrão de $0,5$ minutos. A duração do atendimento prestado pelos caixas também segue uma distribuição normal, com média de 8 minutos para a duração do atendimento e desvio padrão de $1,5$ minutos..
A tela de entrada desta simulação seria:

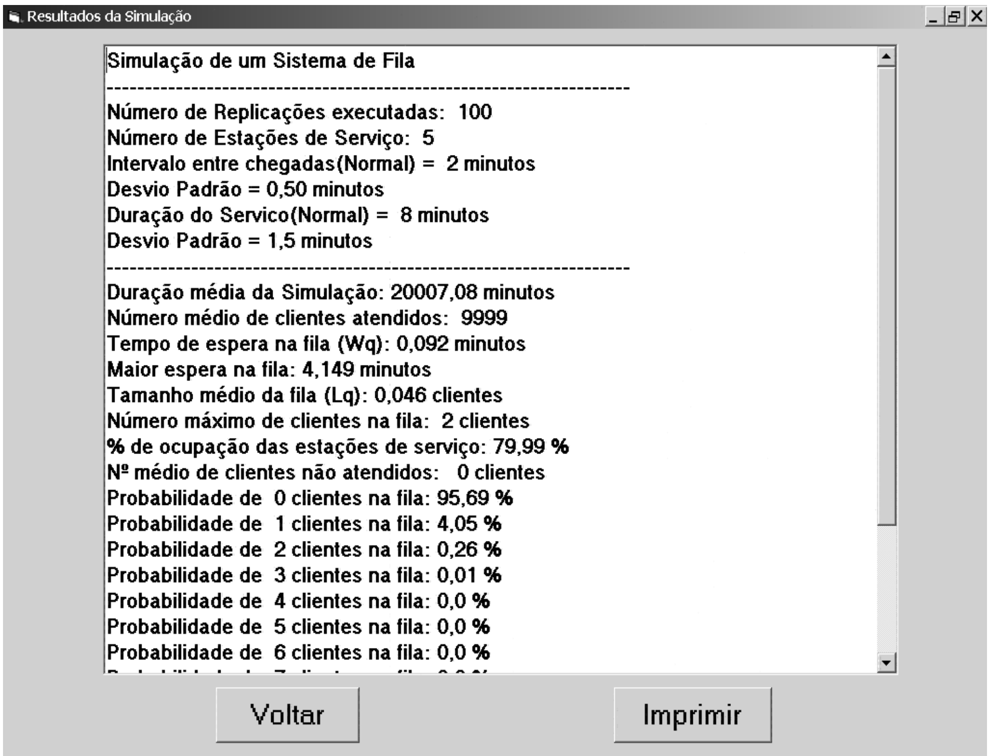


A saída, após 100 replicações, cada uma com 20.000 minutos de simulação, apresentou os seguintes resultados:



Os resultados da simulação mostram que esta agência bancária não está prestando um serviço de boa qualidade pois, em média, cada cliente tem que esperar mais de 33 minutos na fila e, em média a fila tem quase 17 clientes.

Para melhorar a qualidade do serviço prestado vamos examinar os resultados da simulação se colocamos mais 1 caixa de atendimento. Alterando a entrada e executando a simulação, chegamos aos seguintes resultados:



Como pode ser visto, a fila praticamente desaparece.

Embora bastante versátil, o programa é limitado a determinados tipos de sistema de fila. Se alteramos as condições básicas, temos que alterar o programa alterando-se rotinas já existentes ou incluindo novas rotinas.
Como ilustração da dificuldade e do custo de se fazer isto, lembramos que o fonte do programa tem 20 páginas de código de programação. Mesmo se tirarmos as 5 ou 6 páginas de código que são necessários para o ambiente Windows, ainda restariam 15 páginas de codificação!.
Para cada modelo, a não ser que sejam muito semelhantes como nos exemplos acima, temos que construir programas de computador cuja complexidade é proporcional a complexidade do modelo simulado.
Apesar da vantagem advinda de se ter um programa "sob medida", o tempo que se leva para se obter os primeiros resultados normalmente invalida aquela vantagem, principalmente em modelos complexos.
Este fato fez com que fossem desenvolvidos programas voltados exclusivamente para a construção de modelos de simulação. Veremos a seguir, um dos mais usados na atualidade.

5.3 O software ARENA

O ARENA é um software, que roda no ambiente Windows, desenvolvido e de propridade da Rockwell Software Inc [2]. No Brasil é representado pela Paragon Tecnologia Ltda [3].
O ARENA está construído em cima de uma linguagem (SIMAN), própria para simulação, que existe há pelo menos 20 anos. No entanto, como ele roda no ambiente Windows (gráfico), a linguagem fica totalmente transparente podendo-se construir modelos ignorando-se totalmente o que está por trás das telas gráficas.

5.3.1 Obtendo os dados do Modelo

Como já citado anteriormente, para se implementar qualquer modelo de simulação temos que conhecer e tratar os dados de entrada. Vimos, por exemplo, em alguns modelos de filas que partíamos do "enunciado" de que, por exemplo, o intervalo entre chegadas de clientes a uma agência bancária seguia um distribuição exponencial com média de 30 segundos e que a duração média do atendimento pelo caixa era de 20 segundos. É claro, no entanto, que estas frases exigiram muito trabalho para serem formuladas.
Veremos que um dos módulos do ARENA, o *Input Analyzer,* pode ajudar bastante nesta tarefa.

5.3.2 Dados Determinísticos ou Aleatórios

Uma decisão fundamental sobre os dados de entrada de um modelo de simulação, é determinar se eles são determinísticos ou se são variáveis aleatórias seguindo determinada distribuição. Na maioria dos modelos do mundo real, os dados são aleatórios mas um dos erros mais comuns é considerar como constantes, dados que tem comportamento aleatório. É tentador porque é muito mais fácil a análise dos resultados da simulação quando não temos entradas aleatórias o que acarretará também em saídas não aleatórias.
Considerar constante o que é aleatório pode levar a resultados desastrosos como podemos ver no seguinte exemplo: Considere um sistema de fila com uma única estação de serviço. Vamos considerar que o intervalo entre clientes seja <u>exatamente</u> igual a 1 minuto e que a duração do atendimento, pela estação de serviço, seja <u>exatamente</u> igual a 59 segundos. Num sistema deste tipo, a número de médio de clientes na fila será igual a **zero**.
Vamos agora supor que o intervalo entre chegadas tenha média também de 1 minuto mas seguindo uma distribuição exponencial. Idem para a duração do atendimento, ou seja média de 59 segundos seguindo uma distribuição exponencial. Neste caso o número médio de clientes na fila será igual a **58** !.
Escolher o tipo errado de dados pode, por si só, invalidar os resultados de qualquer modelo de simulação.

[2]www.arenasimulation.com
[3]www.paragon.com.br

5.3.3 Coletando dados

A coleta dos dados pode ser uma das tarefas mais demoradas (caras) no desenvolvimento de um modelo de simulação. A qualidade dos dados coletados vai influir diretamente nos resultados que serão obtidos.
Vamos imaginar um modelo que simule uma agência bancária. Se levarmos em conta apenas o sistema de fila dos caixas, dois tipos de dados terão que ser levantados: o do intervalo entre chegadas de clientes ao sistema e a duração do atendimento dos caixas. Para cada um destes eventos será necessário determinar quantos dias de levantamento serão necessários para que os dados coletados sejam uma amostra significativa do processo em questão. Não devemos esquecer que, provavelmente, teremos diferentes tipos de clientes: "normais", preferenciais (idosos, grávidas, etc...), especiais, etc... Cada tipo deste terá "seus" caixas próprios o padrão de chegadas poderá ser bastante diferente entre eles.
No caso dos caixas, a situação é idêntica e também é provável que, para os diferentes tipos de clientes, tenhamos caixas com "velocidades" diferentes, como mais rápidos para os clientes especiais, mais lentos para a fila dos preferenciais e assim por diante.

Na literatura podemos encontrar um grande número de projetos de simulação que fracassaram exatamente por não ter sido feita uma boa coleta dos dados de entrada do modelo. Um dos erros mais comuns é, como a coleta de dados pode ser demorada e cara, dar um "jeitinho", ou seja, simplificar para reduzir custo. Um exemplo seria o caso em que teríamos que coletar dados por 30 dias mas, para economizar, só se coleta por 10 ou 15 dias.

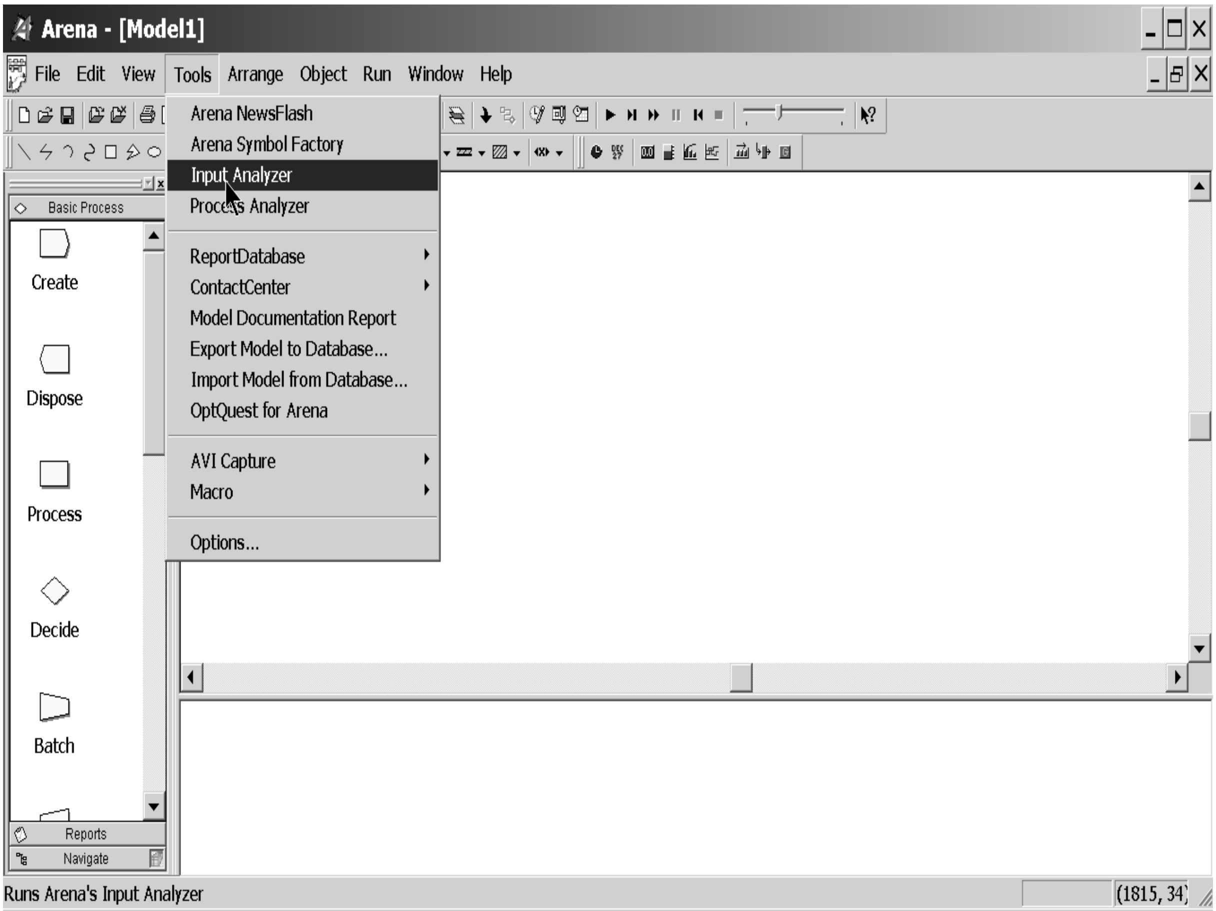
Podemos ter também (raro) a situação ótima qual seja, existirem registros (arquivos históricos) com os dados que precisamos para o modelo. Em muitos sistemas da área industrial, pela própria natureza do trabalho, é comum ter registros do que acontece, por exemplo, nas diversas etapas de uma linha de produção. Estes registros podem permitir que grande parte da etapa de coleta de dados seja desnecessária.

5.3.4 Teste de Aderência com o Input Analyzer

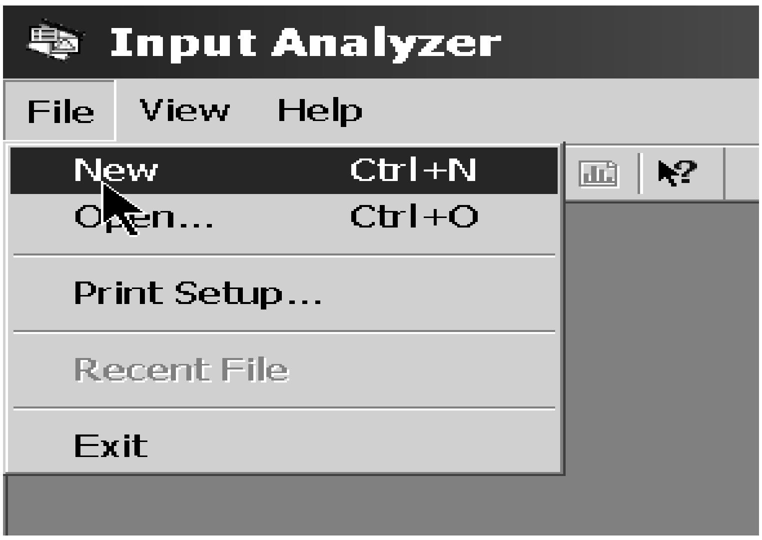
Em um processo de coleta de dados, os 200 valores a seguir foram observados. Como já descrevemos, nosso objetivo agora é determinar se os dados seguem determinado padrão.
Podemos gravar os dados em um arquivo texto que poderá ser lido pelo Input Analyzer. Como podemos ver a seguir, não existe nenhum regra para a estrutura deste arquivo texto. A quantidade de valores em uma linha não é fixa bastando que os valores estejam separados por um branco. Os valores não precisam seguir qualquer ordem e o arquivo pode conter qualquer quantidade de valores.

15.9	15.2	13.9	18.8	15.5	14.8	14.2	
15.6	13.9	16	15.8	15.2	16.1	16.8	
16	14.4	15.2	16.4	16.2	15.8	16.4	
14.3	15.1	16.6	15.7	13.9	17.1	16.7	
17.6	17.1	14.4	18.4	13.7	16.4	15.8	17
14.6	16.8	17	15.5	14	15.8	17	
15	14.3	17.3	17	16	13	16.3	
17.1	14.2	14.9	14.9	14.5	16.6	16.8	
15.9	15.2	16.6	18.5	14.9	16.3	16.6	
16.3	15.9	17.8	15.2	14.6	16.5	14.7	17.5
16.7	16.1	16.8	15.6	17.5	16.9	16.4	15.9
13.9	15.9	16.9	13.7	14.5	14.4	15.1	
16.2	17	15.6	17.3	17.5	15.8	16.8	
15.7	15.7	16.5	14.2	16.6	16.7	15.4	
15.6	18.7	15	14.3	15.3	14.4	14.4	
17.2	16.5	13.2	14.8	15.7	14.3	17.7	
15	14.2	15.7	15.9	13.6	16.3		
14.1	14.4	16.1	15.2	16.5	17.2	15.8	
17.4	16.1	15.1	17.9	14.8	15.9	15.2	
13	14.8	14.1	15.9	16	14.6	17.1	
17.3	14.7	15.9	17.8	15	16.5	14	
17.1	18.5	15.4	15.9	16.9	17.1	17.8	
15.6	15	14.8	16.4	17	17.3	16.3	
17.2	15.8	17.1	14.9	17	16.4		
15.5	15.7	16.1	15.2	14.1	16.5	17.1	14.8
18.4	14.7	16.7	15.1	14.9	16.4	16.1	17.9
17.1	16.3	16.9	15.3	14.9	16.5	16.6	16.1
14.4	16.1	17.6	13.1	18	15.5	14.9	

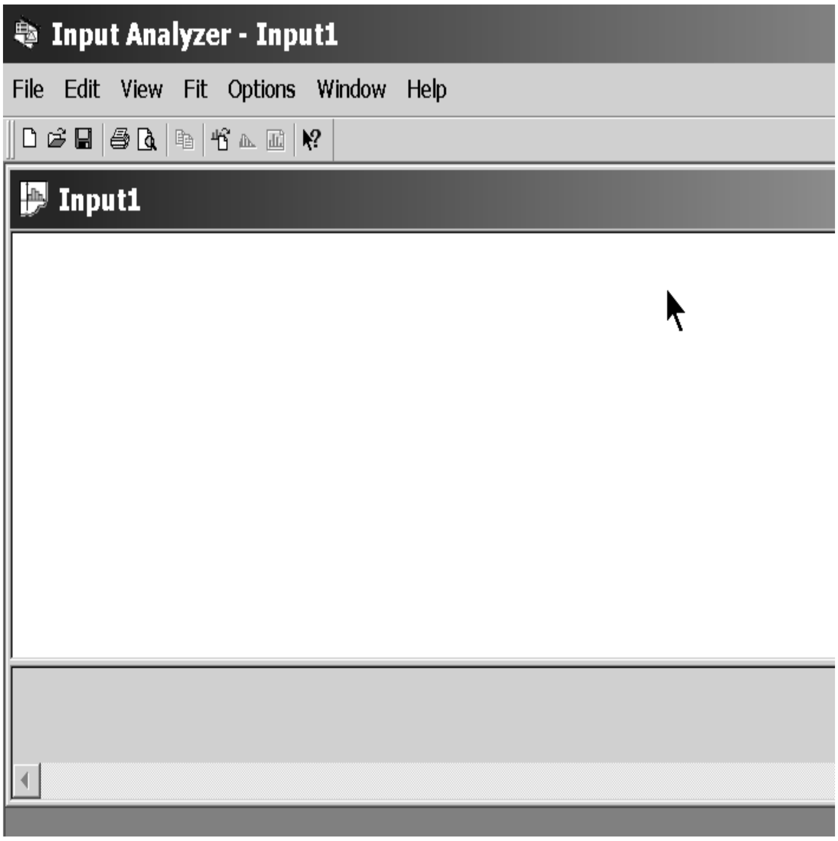
NA tela inicial do ARENA, escolhemos **Tools** e no menu que se abre, **Input Analyzer**:



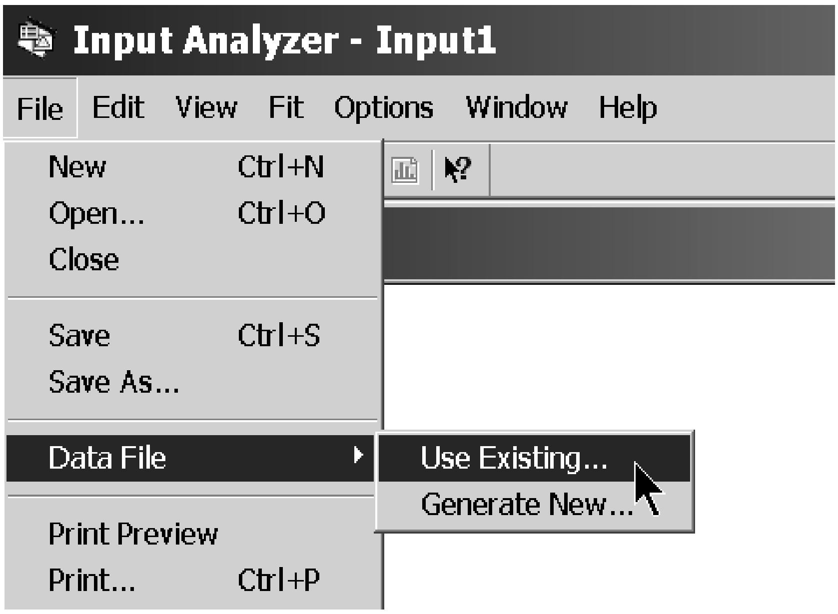
A tela do Input Analyzer vai aparecer e devemos escolher **New**, como abaixo:



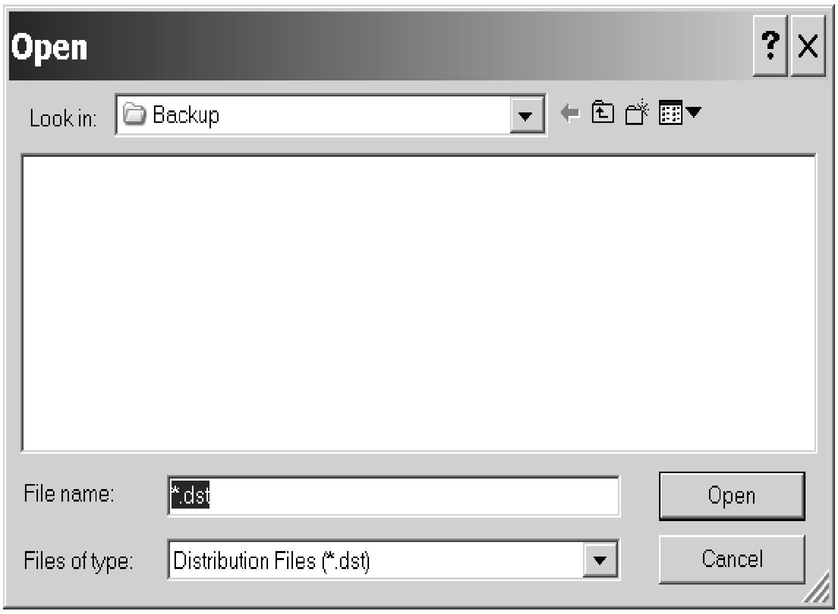
Devemos escolher **File** e **New**, fazendo com que o Input Analyzer mostre a seguinte tela, ou seja a tela de entrada para o programa:



Considerando que os dados estão em um arquivo texto, devemos escolher as opções **File**, **Data File** e **Use Existing**, como podemos ver a seguir.

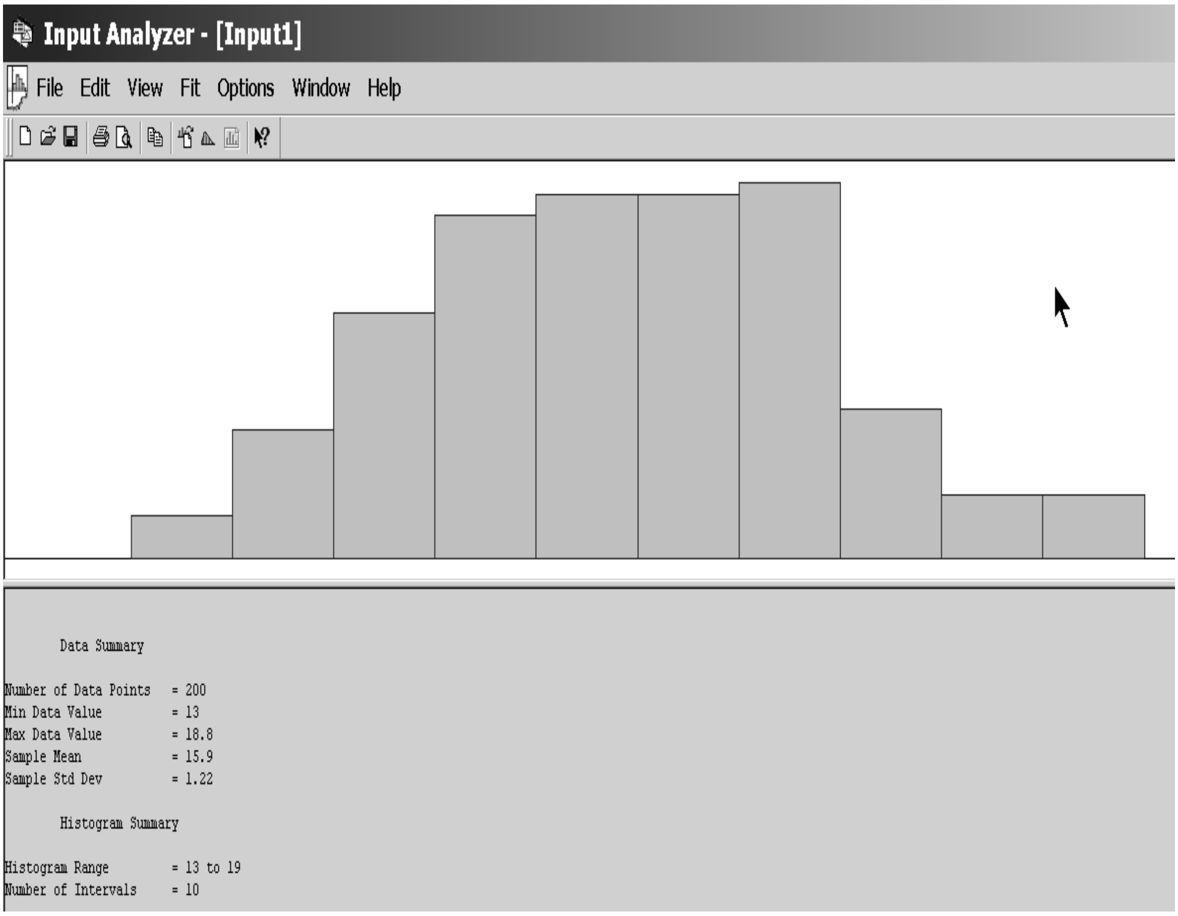


Vai abrir a tela padrão de abertura de arquivo do Windows (**Open**) onde podemos navegar e escolher o arquivo **txt** desejado.

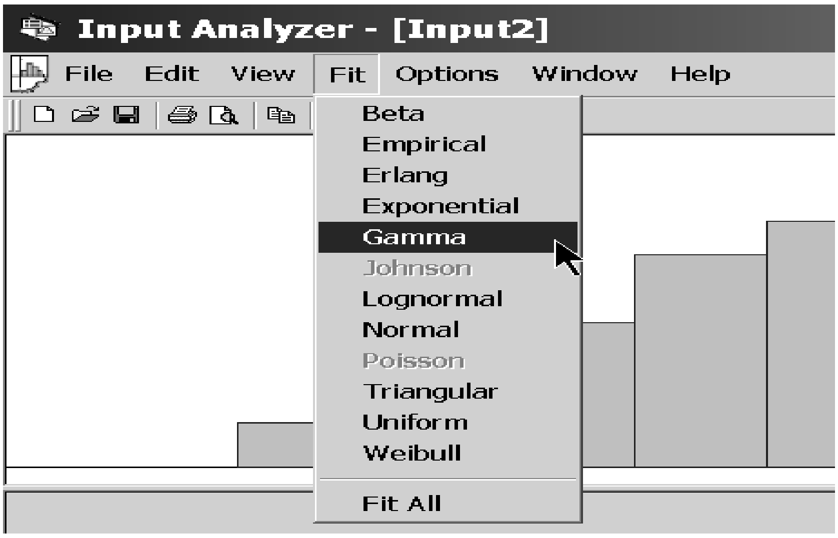


Podemos observar que ele abre procurando arquivos com extensão **.dst** que é o padrão do Input Analyzer. No entanto podemos alterar no **File of Type** e colocar para **.txt**

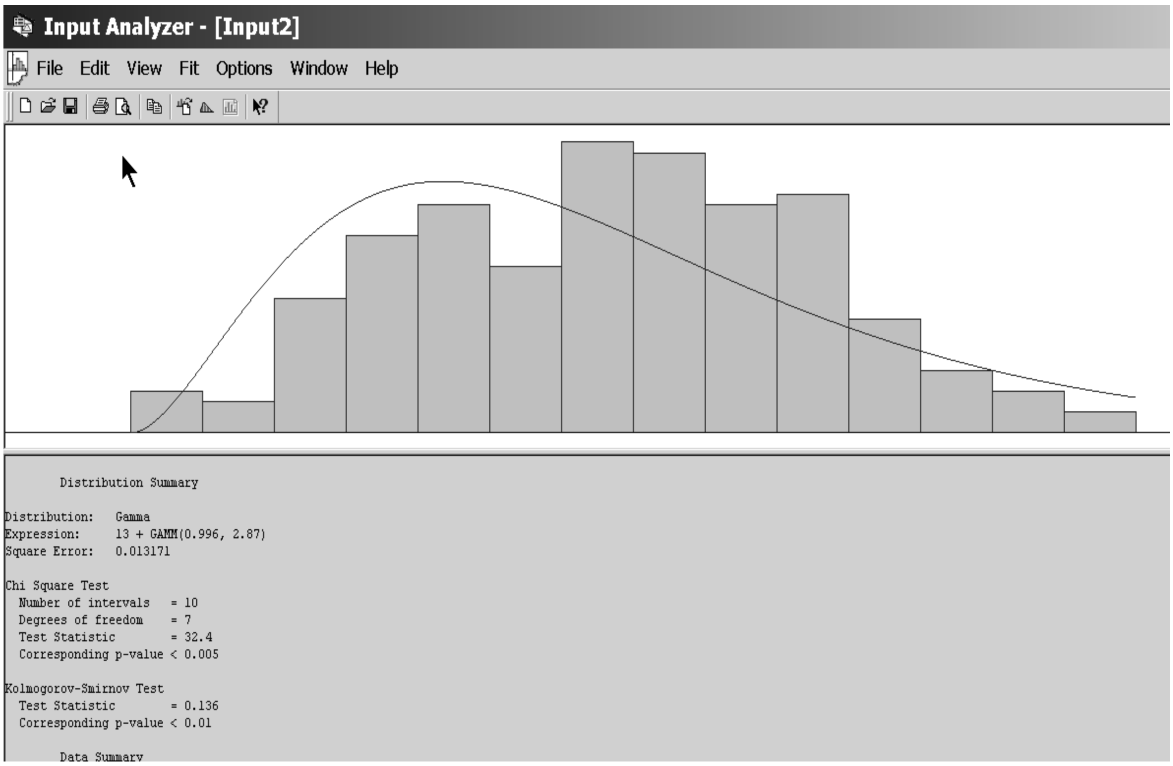
Escolhido o arquivo que contém os dados, o Input analyzer mostra a seguinte tela:



É mostrado o histograma dos 200 valores do arquivo, assim como o valor mínimo, máximo, a média e o desvio padrão. Também é informado os parâmetros da montagem do histograma, ou seja, a faixa que ele mostra e o número de intervalos.
O número de intervalos pode ser alterado pelo usuário e vai influir na própria aderência que o programa faz, como veremos mais adiante. Neste exemplo o número de intervalos foi alterado para 10.
Neste momento podemos fazer a aderência dos dados usando o programa. Vamos supor que, a partir da análise da forma do histograma, decida-se de que uma distribuição Gamma seja a melhor opção para "representar" os dados da amostra. Podemos pedir ao programa para que ele escolha a Gamma com a melhor aderência aos dados, como pode ser visto na tela a seguir, clicando em **Fit** e **Gamma**:

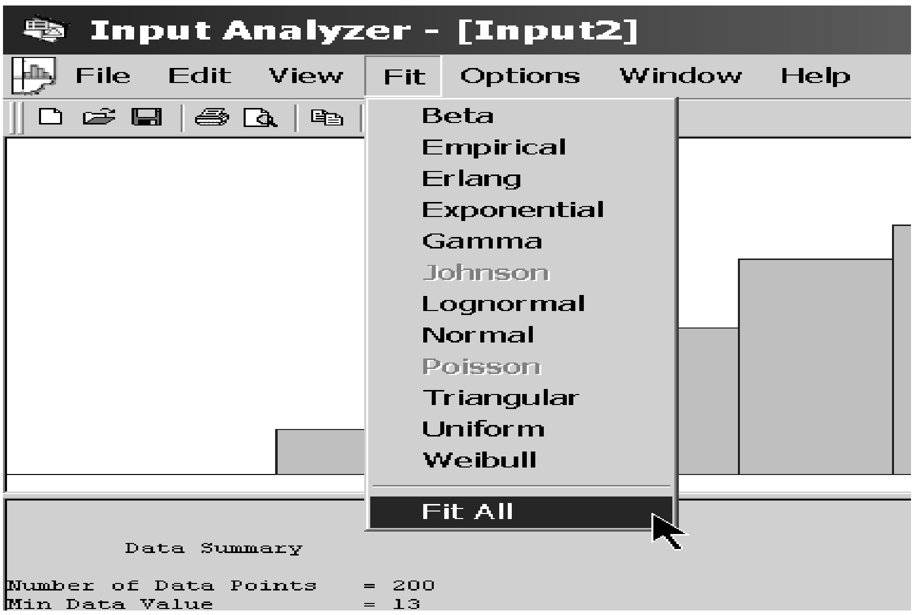


Como resposta, o Input Analyzer mostra a seguinte tela:

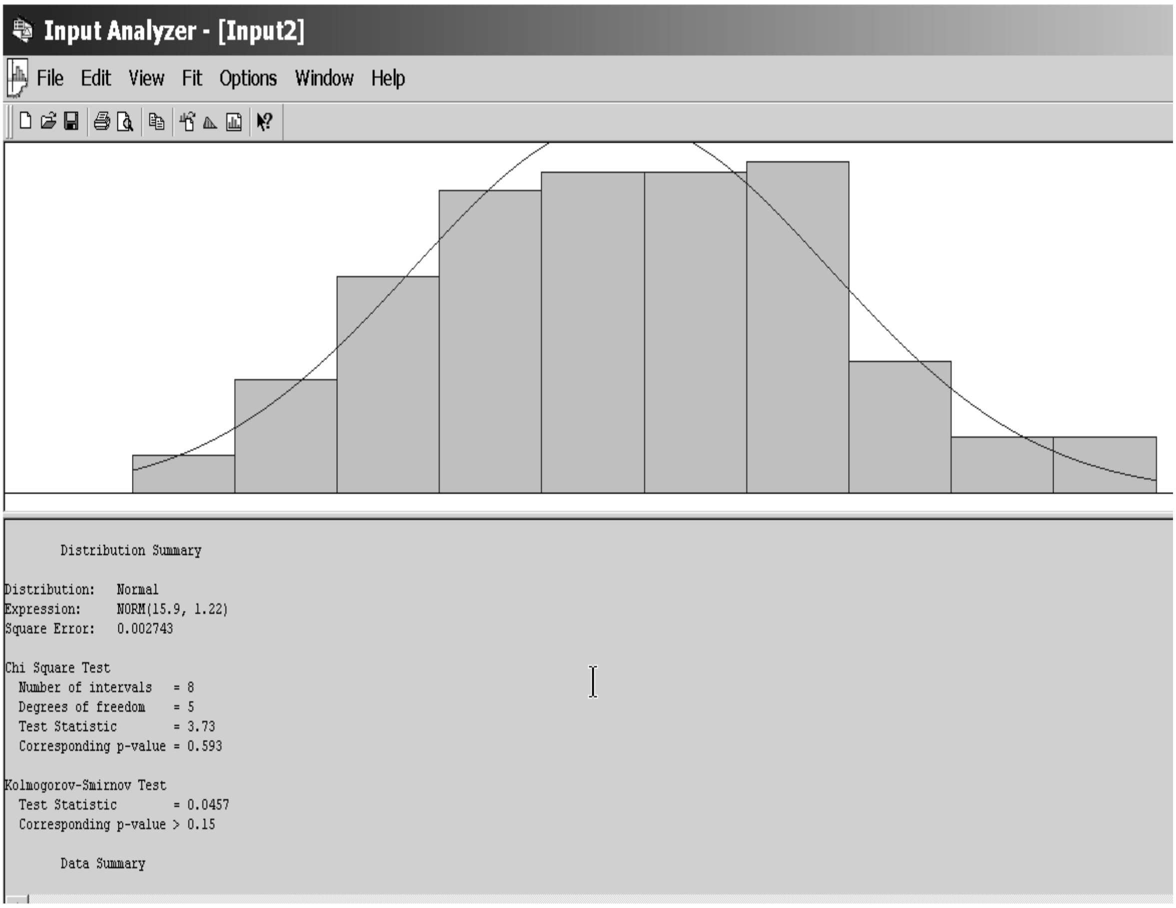


A "melhor" Gamma tem parâmetros $\alpha = 0.996$ e $\beta = 2.87$, com deslocamento de 13.

Mas será que esta escolha é a melhor ?
Podemos ver que o erro, ao quadrado, é igual a 0.016144. Ele é a média entre os erros de cada faixa do histograma. O erro de cada faixa do histograma é o quadrado das diferenças entre as freqüências relativas das observações da faixa e a freqüência relativa da função de distribuição no intervalo da faixa. Logicamente, quanto menor este valor mais a distribuição teórica adere aos dados da amostra
O programa também usa 2 "estatísticas": o teste do χ^2 e o teste de Kolmogorov-Smirnov (KS). O resultado destes testes, como podemos ver acima, são mostrados em função do chamado *p–value*.
O *p–value*, cujo valor está entre 0 e 1, dá a probabilidade do erro cometido caso se rejeite a hipótese de que a distribuição adere aos dados da amostra. Quanto maior o *p–value*, melhor a aderência pois estaríamos cometendo um erro "grande" em não aceitar a distribuição. A regra básica é que os *p–value* devem ser maiores que 0.10 (10%), no mínimo.
Como podemos observar na tela acima, os valores do *p–value* são menor que 0.005 (0,5%) no caso do χ^2 e menor que 0.01 (1%) no caso do teste do KS.
Sendo assim, temos forte indicação de que a Gamma não adere aos dados da amostra.
Qual será então a melhor distribuição para os nossos dados ? Podemos usar o Input Analyzer para responder a esta questão. Escolhemos **Fit** e **Fit All**, como podemos ver abaixo:



Nesta opção o programa vai escolher a melhor distribuição que se ajusta aos dados da amostra. O programa mostra a seguinte tela:



A distribuição que melhor se ajusta é uma Normal com média igual a 15.9 e desvio padrão igual a 1.22. Podemos observar que os *p–value* são iguais a 0.593 (59,3%) e maior que 0.15 (> 15%). O programa também grava um arquivo (summary) em que mostra o *ranking* das diversas distribuições em função do erro quadrado, como podemos ver a seguir:

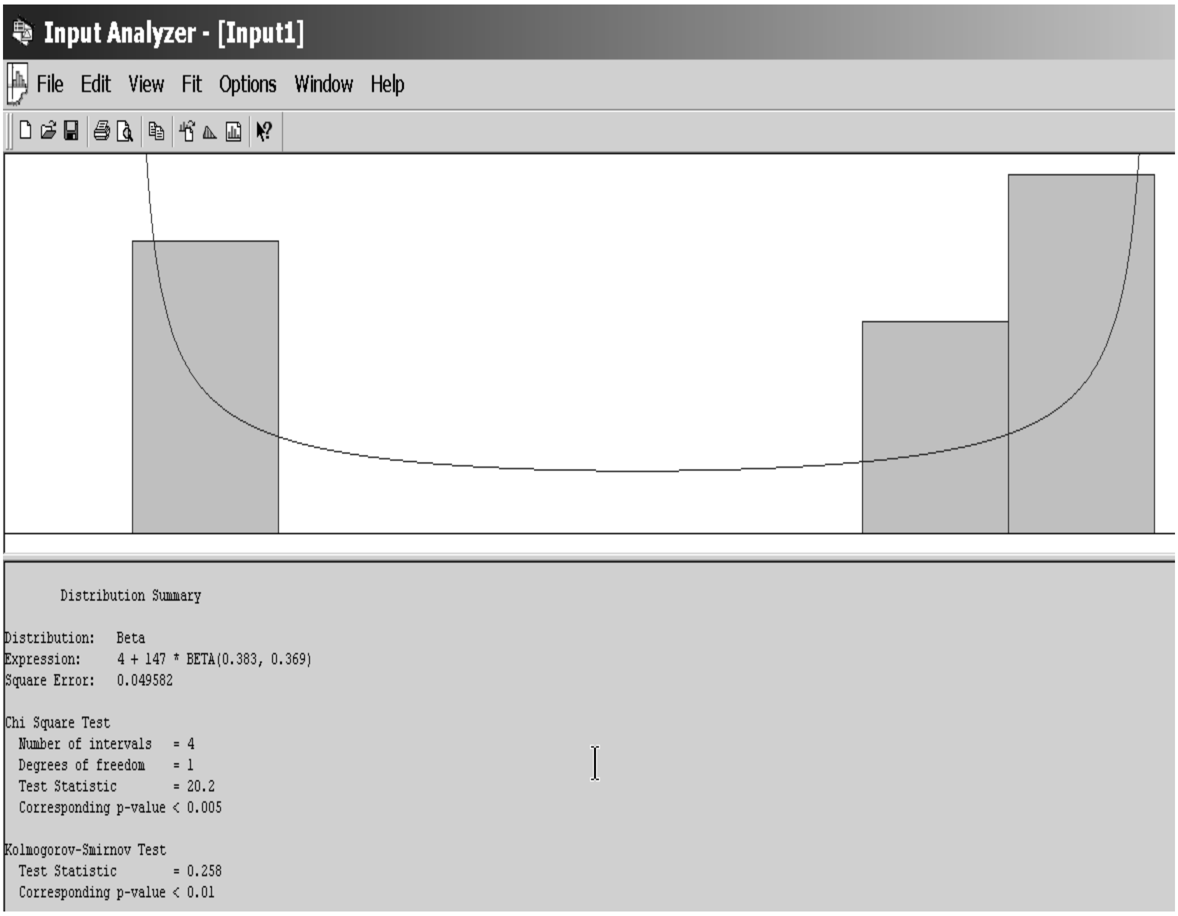
Distribuição	**Erro quadrado**
Normal	0.00274
Beta	0.00328
Triangular	0.00431
Weibull	0.00524
Erlang	0.0151
Gamma	0.0161
Uniform	0.0379
Lognormal	0.0465
Exponential	0.0748

E quando nenhuma distribuição adere aos dados ?

Vamos considerar os dados abaixo:

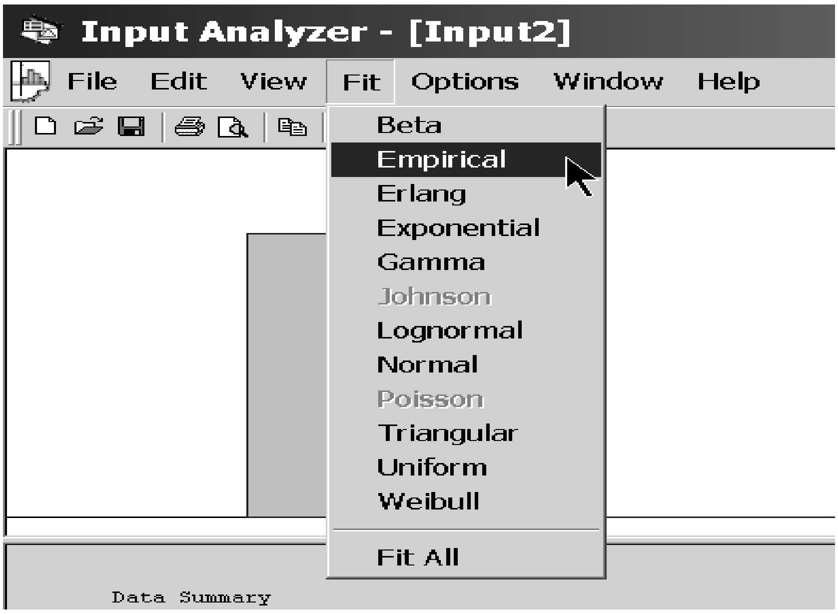
```
120.3  110.8  130.4  140.7  150.7  140.9  150.1
110.5  120.7  130.1  140.2  150.4  110.9  130.3
120.1  150.1  130.7  140.9  150.0  110.9  110.3
120.6  130.9  140.3  150.2  140.9  120.0  110.9
140.5  150.4  130.8  130.9  110.7  120.9  140.9
5.6  4.5  6.1  7.2  4.5  6.3  4.9
7.3  5.4  7.0  4.4  6.6  6.5  5.0
4.8  5.8  5.9  6,4  7.5  6.0  5.2
4.1  5.4  5.6  6.7  5.3  4.9  7.2
```

Submetidos ao Input Analyzer, o "melhor" que ele consegue fazer é o que vemos a seguir:

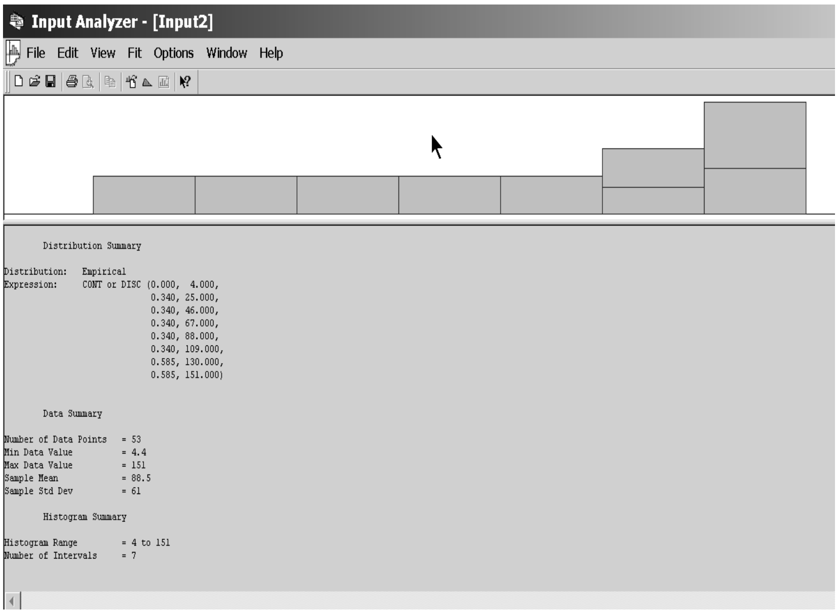


Na verdade, como podemos ver pelos valores dos *p–value*, não foi possível encontrar qualquer distribuição teórica que tivesse uma razoável aderência aos dados da amostra.

Neste caso devemos usar uma distribuição empírica e no Input Analyzer devemos escolher Fit e Empirical, como vemos a seguir:



O programa mostra a seguinte tela:



Podemos ver que ele definiu uma distribuição empírica que pode ser contínua ou discreta:

CONT or DISC (0.000, 4.000,
0.340, 33.400,
0.340, 62.800,
0.340, 92.200,
0.585, 121.600,
0.585, 151.000)

No caso de se usar a distribuição contínua, que parece ser mais adequada aos dados da amostra, o ARENA vai obter valores da distribuição usando interpolação entre os valores das linhas acima.

Não seria melhor dividir ?

O exemplo anterior é um caso típico de 2 grupos distintos de dados: um onde eles variam de 4 a 8 e um outro onde os valores estão entre 110 e 150. Neste tipo de situação, existe uma alternativa melhor do que usar a distribuição empírica abrangendo todo o intervalo: dividir os dados em 2 conjuntos [4 – 8] e [110 – 150]. Para cada conjunto, podemos usar o Input Analyzer para determinar a melhor distribuição. Claro que esta solução vai implicar em se adequar o modelo para que o evento aleatório em questão, seja representado por 2 distribuições distintas.

E os dados "fora da curva" ?

Vamos examinar os dados a seguir:

12.5	17.3	15.1	14.0	12.7	13.9
18.4	16.7	19.1	11.3	17.5	137.6
14.4	16.7	13.8	16.9	14.5	18.1

Os valores estão entre 11 e 19 mas aparece o valor de 137.6, muito diferente dos demais.
Numa amostra, este tipo de valores são chamados de ***outliers*** e, sua presença, pode impedir que se consiga uma distribuição com boa aderência para os dados da amostra.
Quando temos pontos deste tipo, a 1ª providência é verificar se não se trata de um erro na coleta de dados. É muito comum, mesmo quando os dados estão gravados em meio magnético, que alguns valores errados estejam infiltrados na amostra.
Também não se deve cair na tentação de se eliminar os *outliers* sem que antes se faça uma análise sobre a pertinência ou não daqueles valores na amostra.
Se houver dúvidas da validade do dado, uma solução é se aumentar o tamanho da amostra para verificar se a incidência de *outliers* permanece.

Usando os dados históricos *(Flat Files)*

Podemos ter a situação em um modelo de simulação em que determinado processo aleatório como, por exemplo, as chegadas de navios a determinado porto, não vai sofrer grandes alterações ou ser impactado de alguma forma pelo modelo que está sendo construído.
Vamos supor que todas as chegadas de navios a este porto nos últimos 2 anos estejam registradas e armazenadas em um arquivo, ou seja, em meio magnético.
Neste caso em vez de se tentar conseguir uma distribuição que espelhe o comportamento da chegada dos navios, podemos usar, no modelo, o próprio arquivo das chegadas dos últimos 2 anos como *input* no ARENA. Este tipo de arquivo é chamado de *Flat File* e todos os pacotes aceitam este tipo de entrada.

E quando não se tem dados ?

O que descrevemos até agora é a situação em que está sendo construído um modelo para algo que já exista como, por exemplo, uma agência bancária que já funciona ou uma agência nova mas para a qual podemos usar dados de outra agência cujo tamanho, perfil dos clientes, etc..., seja muito semelhante a que está sendo estudada.
Podemos ter, no entanto, o caso em que se está modelando algo, totalmente ou em grande parte, novo. Nestes casos, é possível que não se tenha como obter dados para os processos aleatórios que o modelo eventualmente possa ter.
Neste tipo de modelo, como teremos que inferir como será o comportamento dos dados, será necessário se avaliar, com muito cuidado, os resultados da simulação. Esta avaliação levará, na maioria dos casos, a se rever os dados de entrada voltando-se ao estágio de avaliação dos resultados. É um processo de refinamento até se ter resultados que sejam coerentes com o processo sendo modelado.
Na ausência de dados, se formos usar uma distribuição, devemos "olhar" inicialmente para as distribuições Uniforme, Triangular, Normal e Exponencial. Os parâmetros para estas distribuições são fáceis de entender e tem um conjunto de características que se adaptam a muitos tipos de modelos, como podemos ver na tabela a seguir:

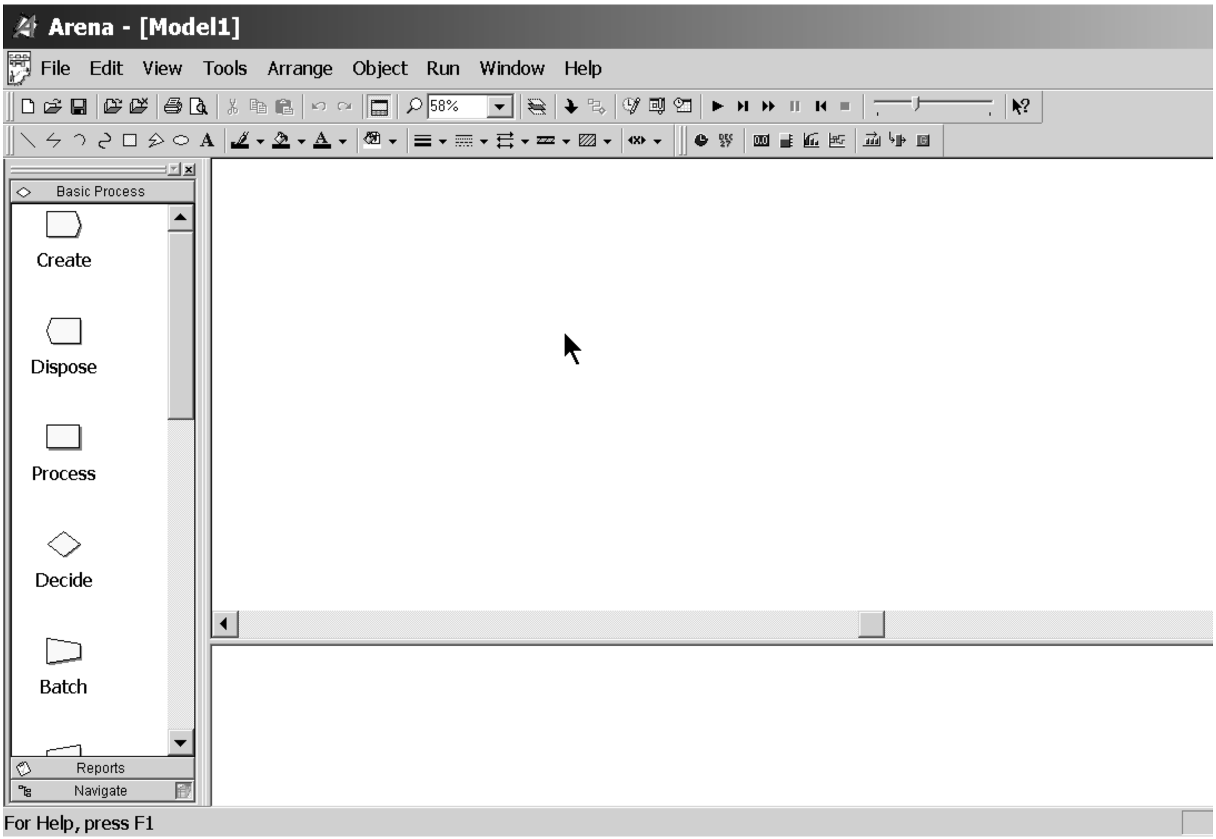
Distribuição	Parâmetros	Características	Exemplo de uso
Exponencial	Média	Variabilidade ampla Limitada à esquerda Ilimitada à direita	Intervalo entre chegadas Tempo entre quebras
Triangular	Mín, Moda, Máx	Simétrica ou não simétrica Limitada em ambos os lados	Duração de atividades
Uniforme	Mín, Máx	Todos os valores com mesma prob. Limitada em ambos os lados	Pouco conhecimento sobre o processo
Normal	Média e Desvio Padrão	Simétrica Variabilidade controlada pelo desvio padrão	Processos que são soma de outros processos Processos com simetria em relação a média

Se os intervalos de tempo são independentes, ou seja, um valor não influencia o próximo, a média tem valor baixo e existe uma grande variabilidade nos tempos, a distribuição **Exponencial** pode ser uma boa escolha. Normalmente é usada para descrever intervalos entre chegadas (à agências bancárias, supermercados, restaurantes, etc...) e durações de atendimento (de caixas de bancos, supermercados, etc...).
Se o tempo representa a duração de uma atividade onde existe um valor mais provável e alguma variação em torno deste valor, a distribuição **Triangular** é freqüentemente usada. Ela é definida pelos valores mínimo, mais provável (moda) e máximo, o que é uma forma natural de se prever a duração de alguma atividade. Ela ainda tem a vantagem adicional de permitir valores não simétricos em torno do valor modal. Um ponto fraco da distribuição Triangular é que ela é uma distribuição limitada em ambos os lados, ou seja, não se pode ter nenhum valor menor que o mínimo nem maior que o máximo.
Quando não sabemos praticamente nada sobre determinado processo mas sabe-

mos, mesmo que seja aproximadamente, o valor mínimo e máximo, a distribuição **Uniforme** pode ser uma opção. Ela fornecerá qualquer valor, entre os 2 extremos, com a mesma probabilidade.
Se temos valores que são simétricamente distribuídos em torno da média e sem limite inferior e superior, a distribuição **Normal** pode ser uma boa escolha. Um problema que temos com a distribuição Normal é que, como estamos normalmente falando de tempo, não podemos ter valores negativos. Nenhuma atividade pode durar, por exemplo, – 1 minuto. Assim se temos média igual a 4 minutos e desvio padrão igual a 1 minuto, é provável que durante a simulação apareçam valores negativos. O ARENA transforma todos estes valores negativos em zero. Alguns outros softwares, desprezam eventuais valores negativos.
Se a média está 3 ou 4 desvios padrões acima de zero, é provável que a distribuição Normal não seja a mais apropriada para o processo em questão.

5.3.5 O uso do ARENA

A tela principal do ARENA é a seguinte:



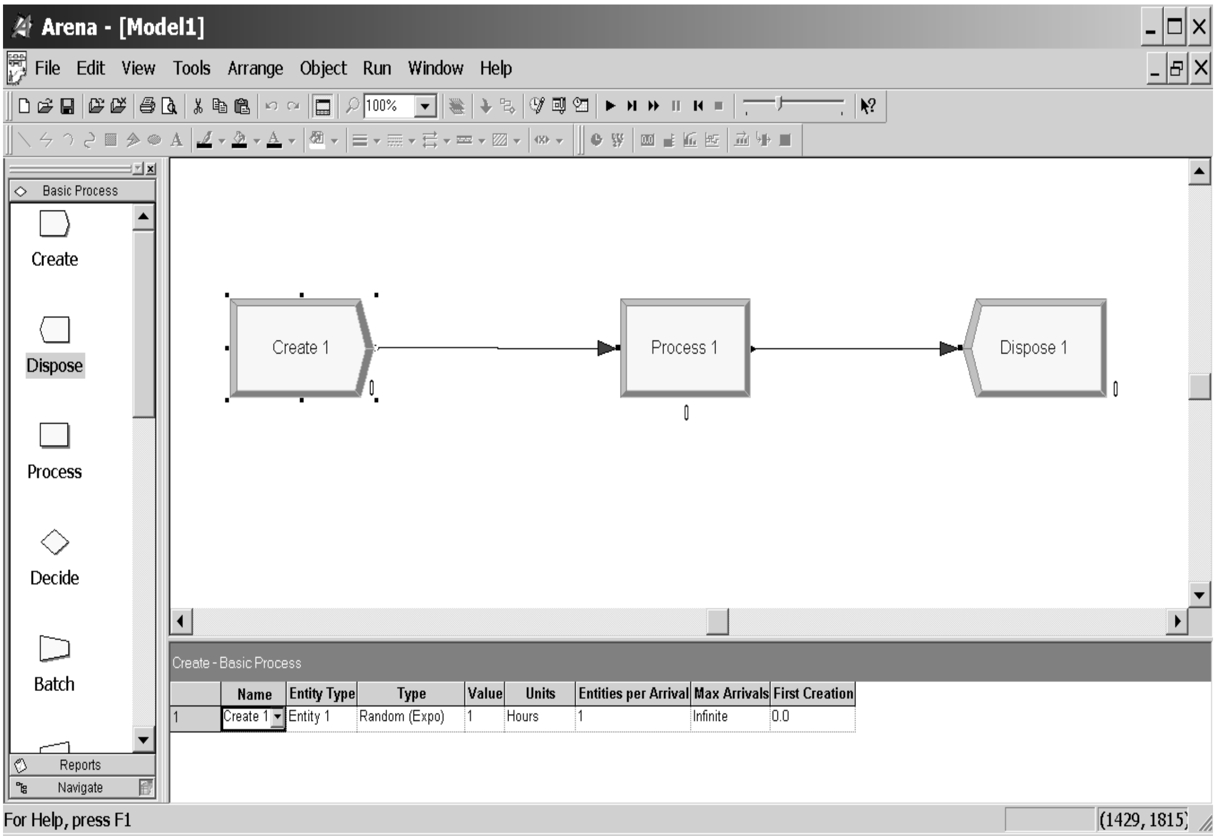
A esquerda temos a área dos painéis (*templates*) como Create, Dispose, Process, etc... Os modelos de simulação são construídos com estes painéis, como veremos mais adiante. Observe que o título da área de painéis é "Basic Process", ou seja

estes são os blocos básicos e poderemos acrescentar outros a medida que eles forem sendo necessários.
Na verdade, na sua forma mais simplificada, criamos modelos no ARENA construindo “desenhos” lógicos com os painéis.

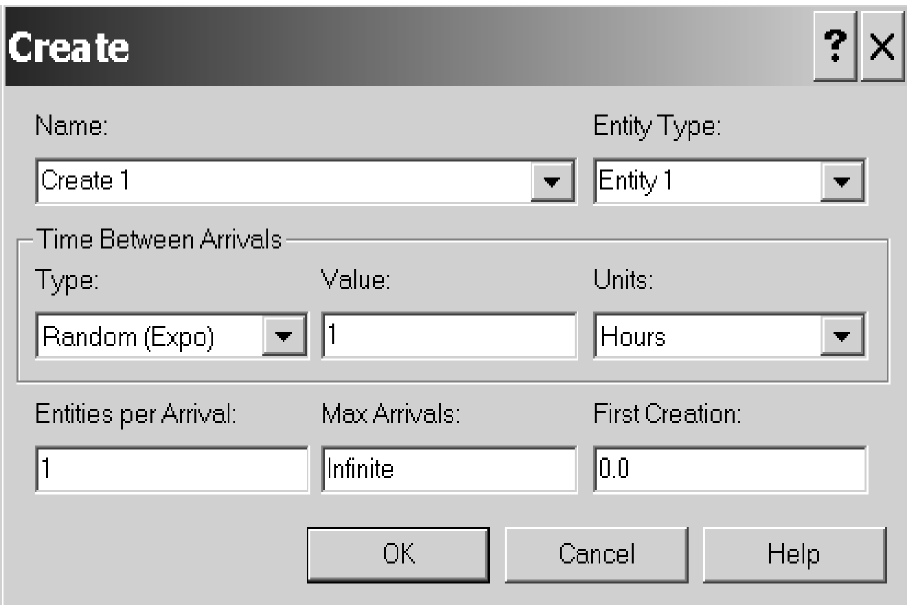
5.3.6 Simulação de um pequeno posto bancário

Vamos ver o exemplo de um pequeno posto bancário com um único caixa.
Como já dissemos, no Arena trabalhamos com blocos e, interligando-os, construímos o modelo. Antes de ver como se faz no Arena, vamos construir nosso modelo lógico. Podemos representar nosso posto da seguinte forma:

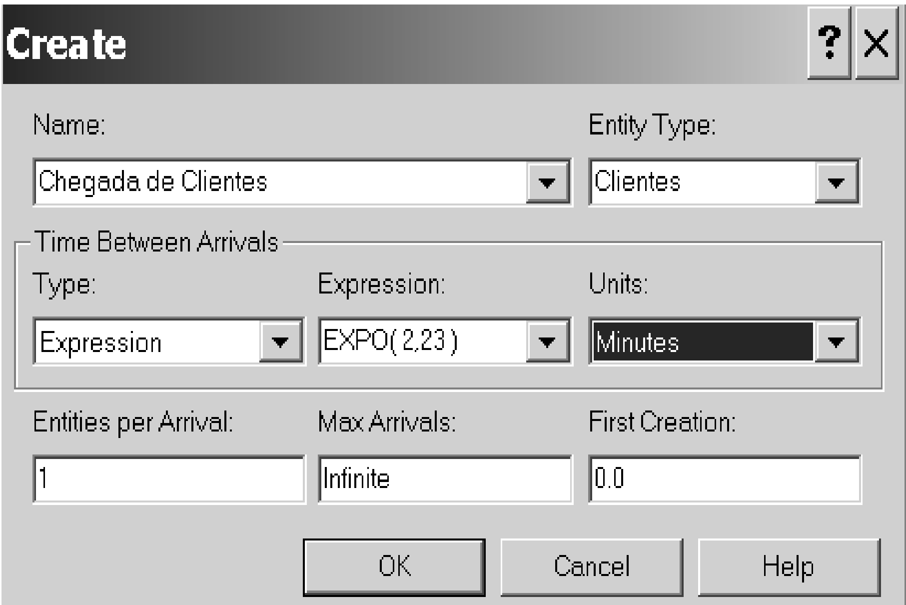
Vamos agora para o Arena. O nosso 1º bloco, ou seja a chegada de clientes, tem que ser criado. Clicando e segurando o mouse, arrastamos o template *Create* do lado esquerdo para a área de trabalho, que é a maior área branca da tela inicial. Soltamos o mouse e uma caixa **Create1** aparece no espaço em branco. A seguir temos que criar o bloco que representa o atendimento aos clientes. Para isto, temos que usar o template *Process*, ou seja clicamos, seguramos o mouse e o arrastamos para a direita da caixa Create 1. O ARENA conecta automaticamente as 2 caixas (isto pode ser desligado). Vai ser criada uma caixa chamada **Process 1**. E, para finalizar, fazemos o mesmo com o template *Dispose* e a colocamos à direita da Process 1. Esta caixa vai representar a saída dos clientes.
A aparência da nossa tela deverá ser a seguinte:



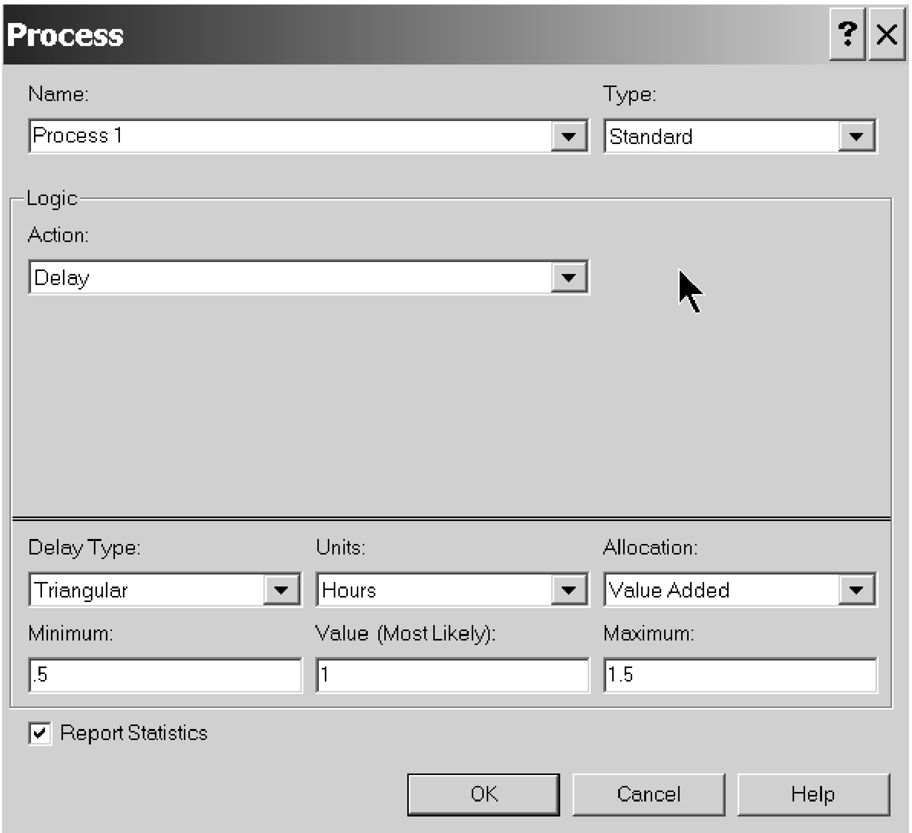
Na parte abaixo da área de trabalho, aparece uma espécie de planilha com alguns dados que ficarão mais claros a medida que formos avançando.
Dando um duplo click na caixa **Create1**, abre-se o seguinte quadro:



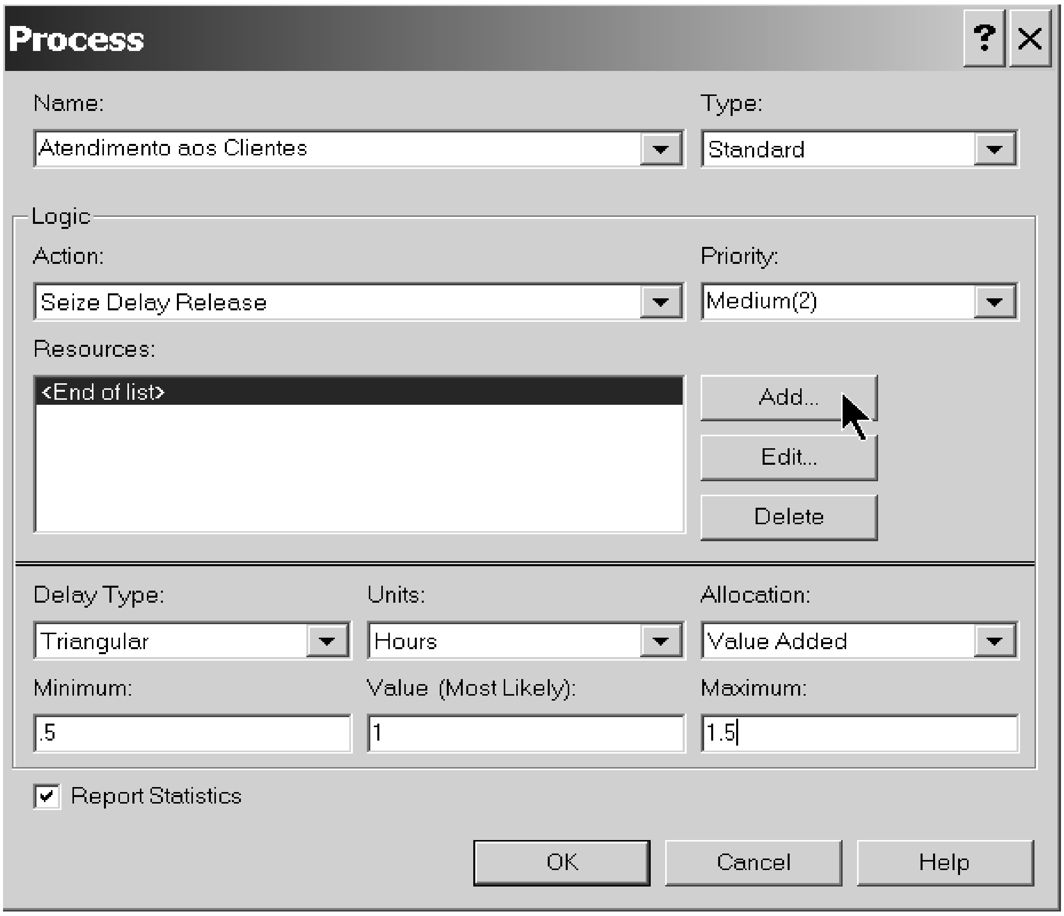
Podemos entrar com os dados da nossa aplicação. Na 1ª linha, vamos trocar o nome (*Name*) de Create 1 para *Chegada de Clientes*. O tipo da entidade (*Entity Type*) de Entity 1 para *Clientes*.
A 2ª linha trata da distribuição do intervalo entre chegadas (*Time Between Arrivals*). Em *Type*, vamos escolher *Expression* e em *Value* escolhemos *EXPO(Mean)*, ou seja estamos dizendo que o intervalo entre chegadas segue uma distribuição exponencial. No exemplo que queremos simular, o intervalo entre chegadas tem média igual a 2 minutos.
Antes de introduzirmos este valor, devemos saber que o ARENA usa a RAND4 (pág. 25) como gerador de números aleatórios. Assim, além da média, convém informar, embora seja opcional, qual a série que deve ser usada para gerar as chegadas de clientes. Em princípio não deve-se usar a série 10 pois esta é a série que o Arena usa quando não se informa nada. Vamos usar a série 23. Colocamos então no campo *Value* a expressão EXPO(2,23), ou seja média igual a 2 usando-se, na simulação, a série 23 da RAND4. A seguir devemos trocar o campo *Units* para minutos.
A 3ª linha fica como está pois Entidades por Chegada (*Entities per Arrival*) é igual a 1, ou seja cada cliente chega sozinho. Não existe limite para o número de clientes que podem chegar, assim *Max Arrivals* é igual a Infinito e o instante de criação (*First Creation*) da 1ª chegada é o instante 0.0, ou seja o início da simulação.
Nosso quadro fica então desta forma:



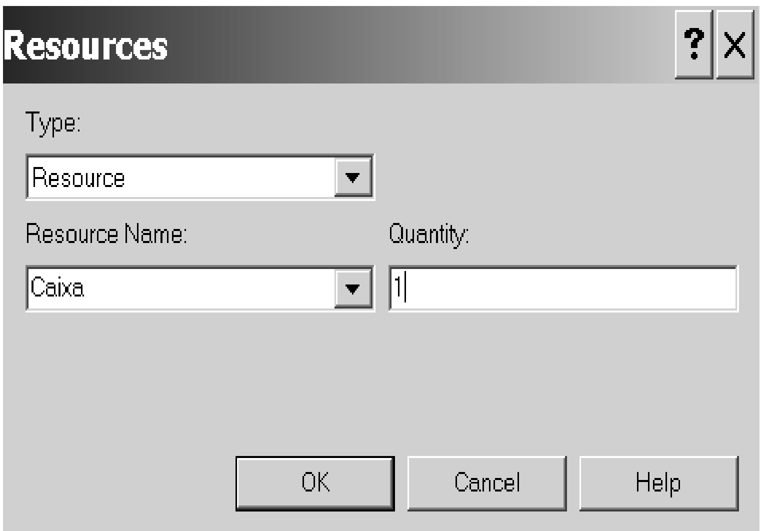
Basta clicar em **OK** para confirmar as alterações.
Vamos agora dar um <u>duplo click</u> no bloco **Process 1**. Um novo quadro se abre:



Mudamos o *Name* para Atendimento aos Clientes, o tipo fica Standard. No campo *Action*, deve ser escolhida a opção *Seize Delay Release*, ou seja Captura–Retém por algum tempo–Libera, que é o que um caixa faz com os clientes.
Ao se escolher a ação acima abre-se um novo campo Prioridade (*Priority*) que deve ser deixado como Medium(2). Abre-se, também um novo quadro para a definição dos recursos (*resources*) que, no nosso exemplo, é um único caixa.

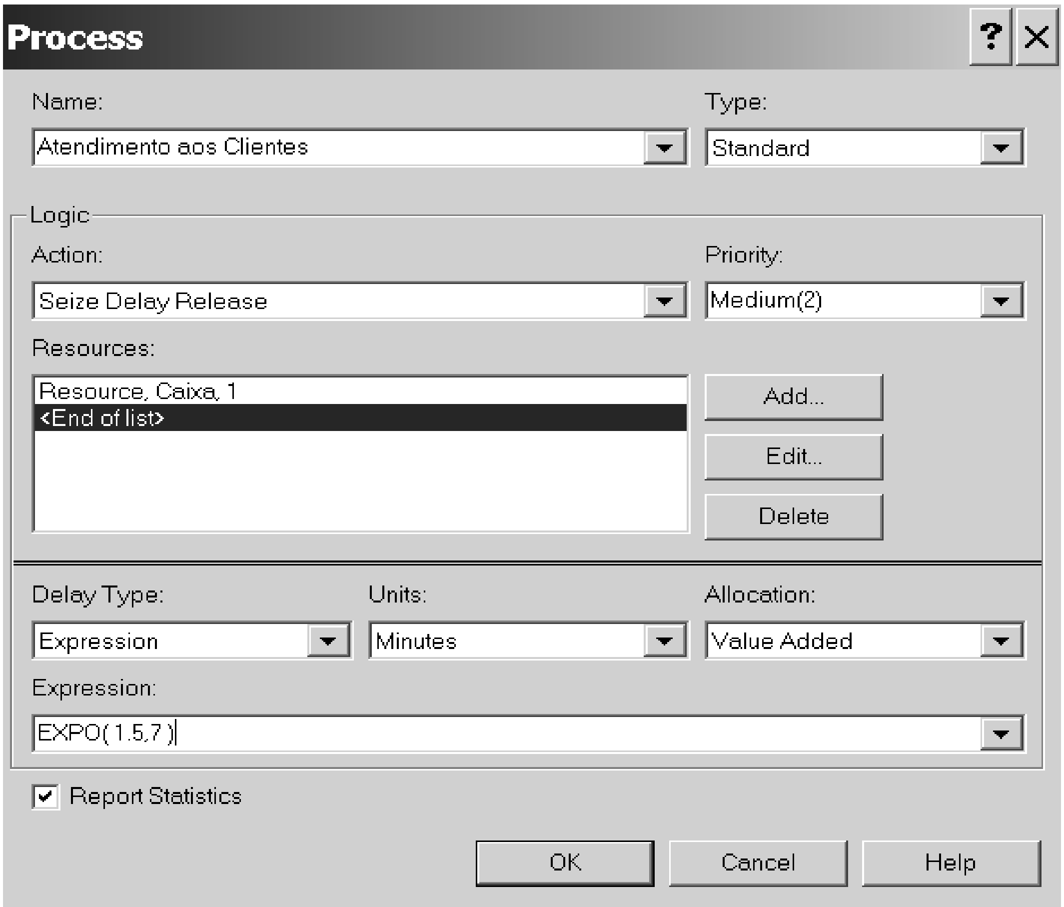


Devemos clicar na opção *Add* para incluir o recurso (Caixa) no modelo, como mostrado a seguir:

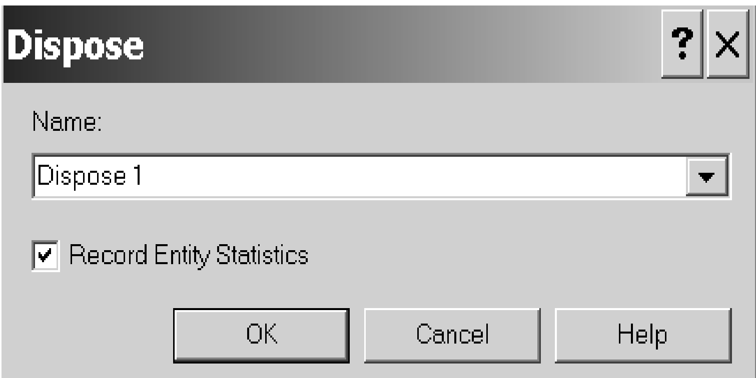


Como pode ser visto acima, escolhemos Resource (em *Type*), Caixa (em *Resource name*) e 1 (em *quantity*).

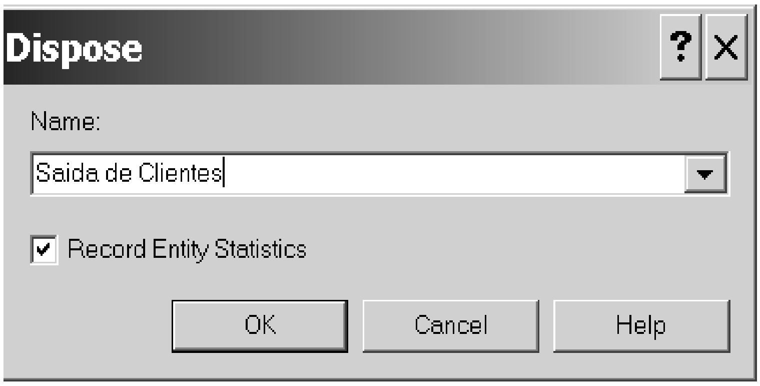
Precisamos informar agora o tipo de distribuição do atendimento feito pelo caixa que, no nosso exemplo, segue uma distribuição exponencial com média de 1.5 minutos. Escolhemos então: Expression (em *Delay Type*), Minutes (em *Units*), Value Added (em *Allocation*) e EXPO(Mean) (em *Expression*). Devemos a seguir substituir o (Mean) por (1.5,7), pois vamos usar a série 7 da RAND4 para a geração da duração do serviço. Se não estiver marcado, devemos marcar o campo *Report Statistics*. Finalmente clicamos em **OK** para confirmar nossos dados.



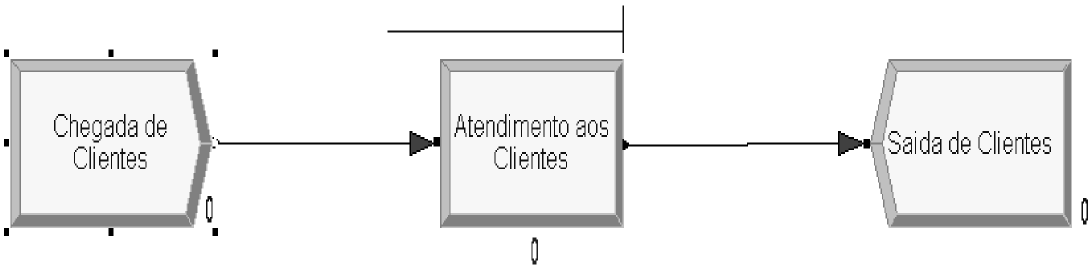
Repetimos o processo para a caixa **Dispose**. O seguinte quadro aparecerá:



Mudamos o nome para Saída de Clientes, marcamos o campo *Record Entity Statistics* e clicamos em **OK**, obtendo-se o seguinte quadro.



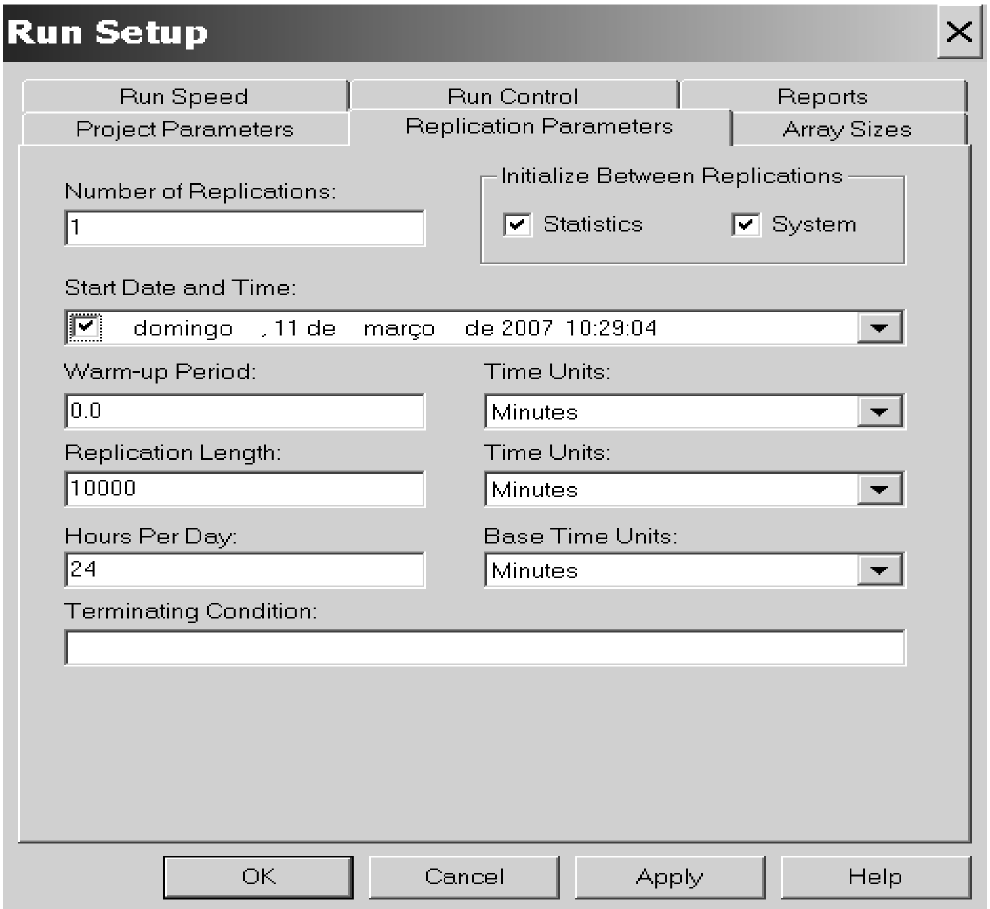
A área de trabalho do ARENA estará da seguinte forma:



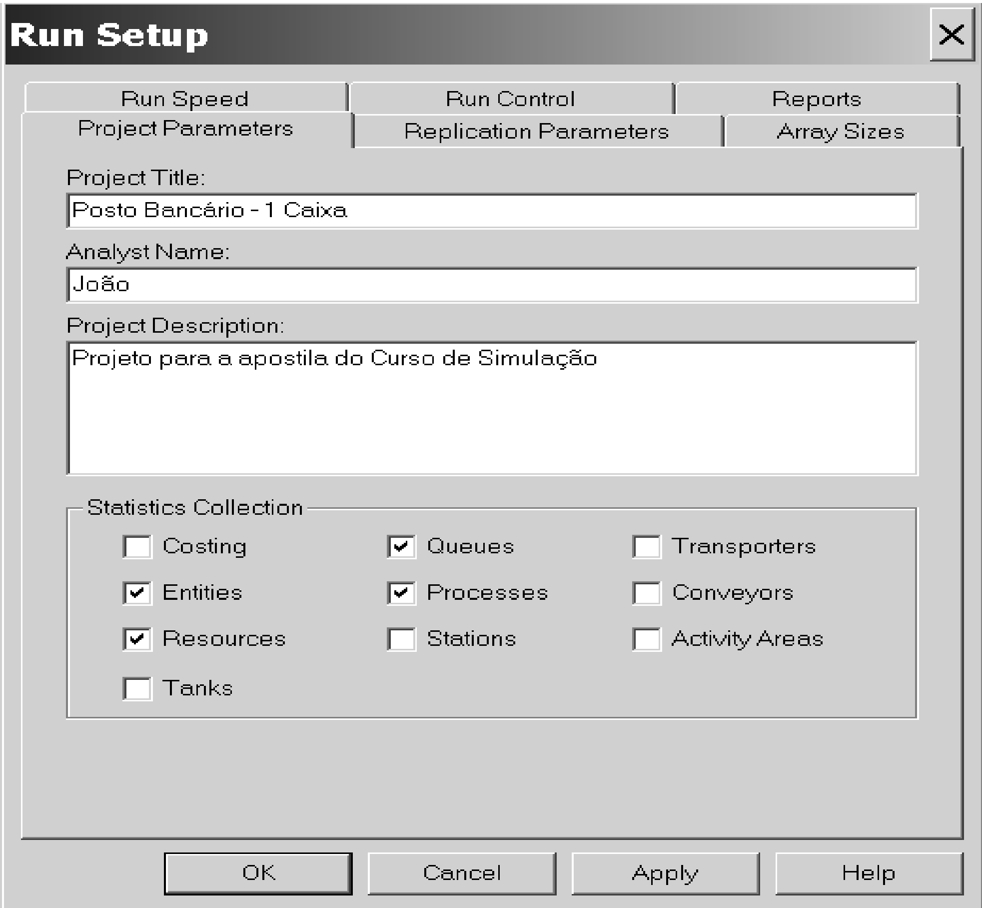
Podemos reparar que aparece uma linha em cima do bloco Atendimento aos Clientes. Esta linha representa a fila que se forma no caixa. Na parte de baixo, a medida que se clica em um dos blocos, aparece uma planilha com as informações pertinentes a cada bloco. Podemos alterar, se for o caso, as informações na planilha que elas passarão a valer para o bloco.

	Name	Entity Type	Type	Expression	Units	Entities per Arrival	Max Arrivals	First Creation
1	Chegada de Clientes	Clientes	Expression	EXPO(2,23)	Minutes	1	Infinite	0.0

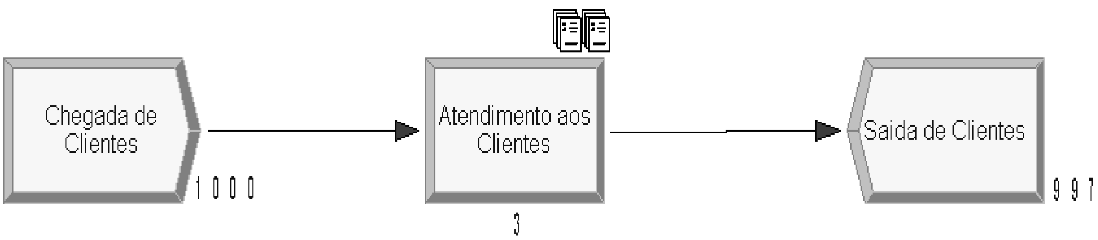
Neste ponto, o modelo está construído mas antes de executar a simulação, temos que ajustar alguns parâmetros. Assim, devemos clicar em **Run** → **Setup**, no menu. O quadro a seguir (já com as alterações feitas), para *Replication Parameters* aparecerá:



Alteramos todas as unidades de tempo (*Time Units*) para minutos e vamos rodar a simulação apenas 1 vez (1 replicação) com duração de 10.000 minutos.
Também no **Run** → **Setup**, aba **Project Parameters**, podemos entrar com os parâmetros do projeto tais como nome do projeto, nome do analista, descrição do projeto, etc...
Devemos também marcar os ítens para os quais queremos que sejam gerados relatórios:



Para finalizar clicamos em **OK**.
Para executar a simulação clicamos, no menu, em **Run** → **Go**. Passamos a ver então a simulação dinamicamente acontecendo como no "retrato" a seguir (obtido apertando-se, durante a simulação, a tecla <ESC>):



Neste ponto, já tinham chegado 1.000 clientes, 997 já tinham sido atendidos, 2 estavam na fila e 1 estava sendo atendido.
Ao final da simulação, o ARENA emite dezenas de relatórios que podem ser acessados a partir da "aba" *Reports* que existe na parte esquerda da tela. Vamos mostrar alguns destes relatórios e ver os resultados obtidos.
O 1º é o *Queues* que dá informações sobre a fila em si:

10:51:20 **Queues** março 11, 2007

Posto Bancário Replications: 1

Replication 1 Start Time: 0,00 Stop Time: 10.000,00 Time Units: Minutes

Atendimento aos Clientes.Queue

Time	Average	Half Width	Minimum	Maximum
Waiting Time	4.5949	1,08266	0	44.6915

Other	Average	Half Width	Minimum	Maximum
Number Waiting	2.2995	0,556066048	0	21.0000

O tempo de espera médio na fila é de 4.59 minutos. O teórico (W_q), modelo M/M/1, é 4.50 minutos. A maior espera foi de 44.69 minutos. O número médio de clientes na fila é 2.3 clientes. O teórico (L_q) é igual a 2.25 clientes. O número máximo de clientes na fila foi 21.

O próximo relatório que podemos examinar é o de *Resources* que, no nosso exemplo é o caixa.

10:54:30 **Resources** março 11, 2007

Posto Bancário Replications: 1

Replication 1 Start Time: 0,00 Stop Time: 10.000,00 Time Units: Minutes

Resource Detail Summary

Usage

	Inst Util	Num Busy	Num Sched	Num Seized	Sched Util
Caixa	0,75	0,75	1,00	5.002,00	0,75

Como podemos observar, o Caixa ficou ocupado 75% do tempo. Este resultado é exatamente igual ao teórico ($\rho = 0.75$). Mostra também que o Caixa atendeu 5.002 clientes.
Outro relatório que pode ser visto é o de *Processes*:

10:59:03 **Processes** março 11, 2007

Posto Bancário Replications: 1

Replication 1 Start Time: 0,00 Stop Time: 10.000,00 Time Units: Minutes

Atendimento aos Clientes

Time per Entity	Average	Half Width	Minimum	Maximum
Total Time Per Entity	6.0954	1,10952	0.00192473	46.8537
Wait Time Per Entity	4.5936	1,08266	0	44.6915
VA Time Per Entity	1.5018	0,048842752	0.00005609	14.7120

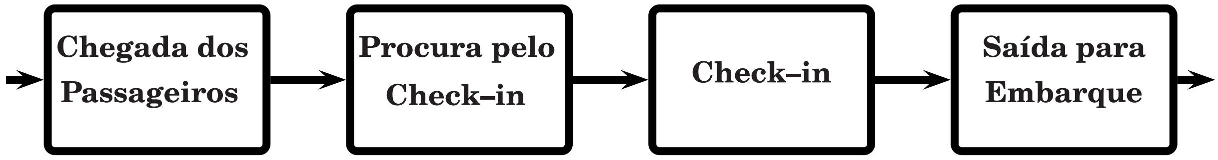
Accumulated Time	Value
Accum Wait Time	22,972.84
Accum VA Time	7,510.42

Ele mostra que a maior espera total de um cliente foi de 46.85 minutos, o atendimento mais demorado foi 14.71 minutos e a maior espera na fila foi de 44.69 minutos.
Como citamos anteriormente, vários outros relatórios, com gráficos inclusive, são impressos e podem ser examinados.

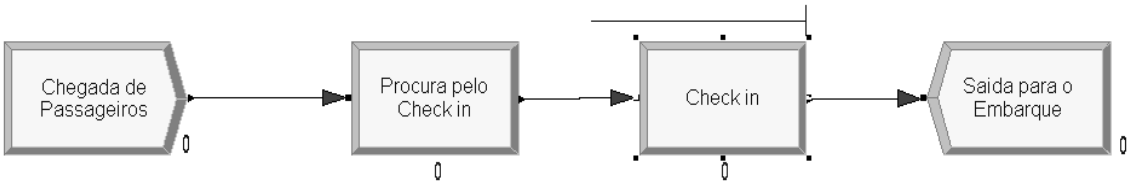
5.3.7 Simulação de um check-in

Vamos imaginar a seguinte situação: passageiros, que vão viajar em determinada companhia aérea, chegam ao aeroporto com intervalo médio de 1.6 minutos, de acordo com uma distribuição exponencial. Estes viajantes, desde do momento em que entram no aeroporto até chegar ao balcão de check-in, demoram entre 2 e 3 minutos com distribuição uniforme entre estes 2 valores. No check-in, os viajantes tem que entrar em uma fila única até que um dos 5 atendentes o atenda.O tempo no check-in segue uma Distribuição de Weibull com parâmetros $\alpha = 3.91$ e $\beta = 7.76$. Após o check-in, eles podem se dirigir aos portões de embarque. Visando melhorar o atendimento, deseja-se determinar uma série de variáveis tais como o tempo médio

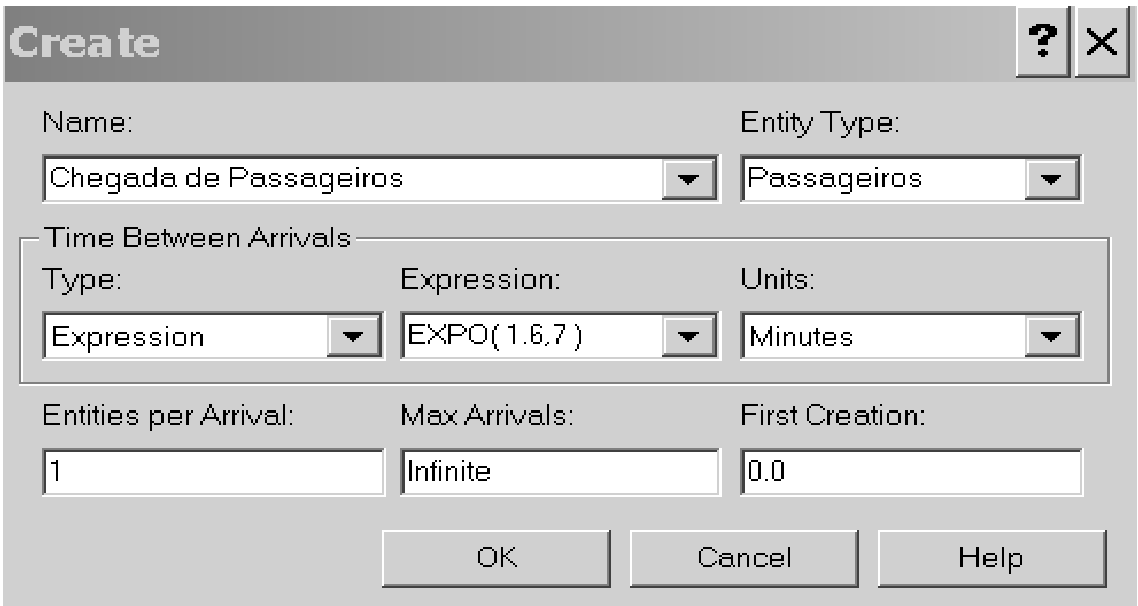
que um passageiro gasta desde de que chega no aeroporto até ser liberado para o embarque, o número médio de viajantes na fila do check-in, o nº de passageiros atendidos no check-in de 8:00 horas da manhã até 22:00 horas, etc...
O modelo desta situação pode ser representado pelo gráfico a seguir:



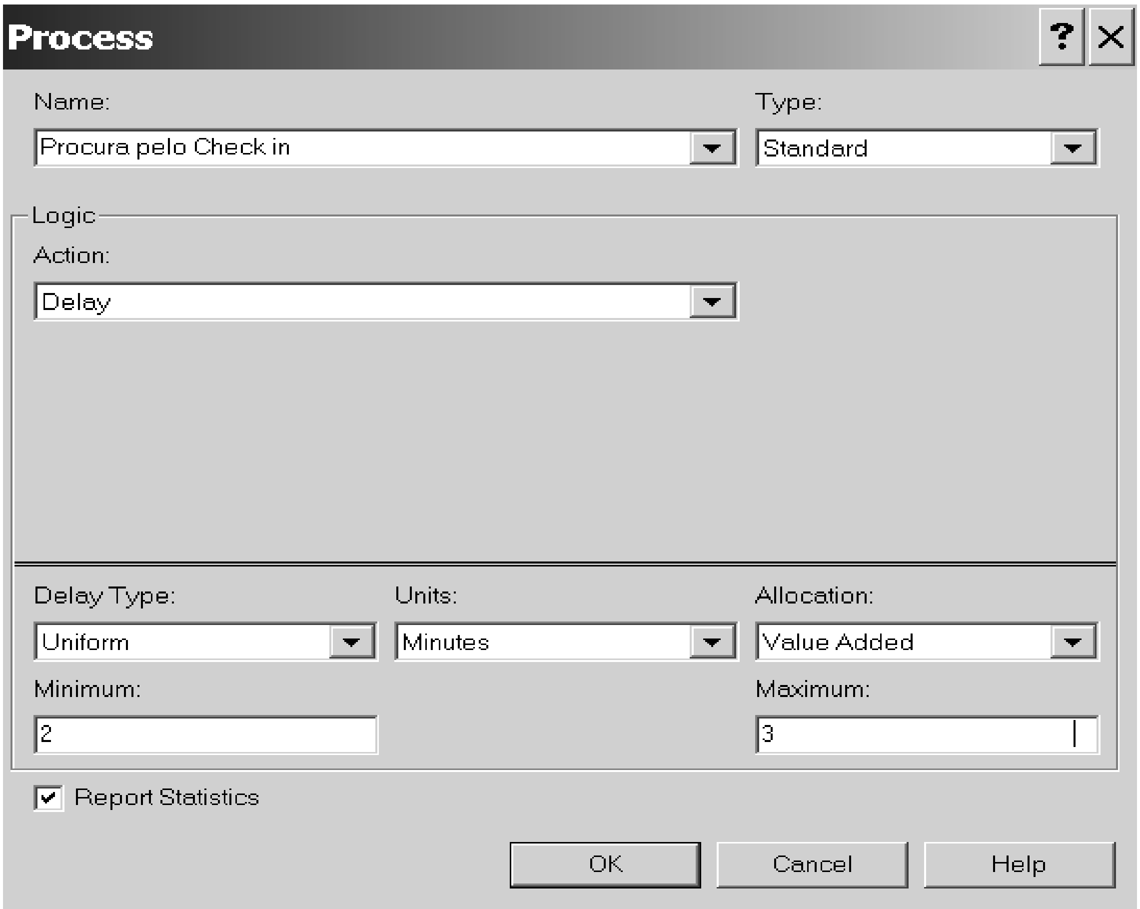
Como fizemos no 1º exemplo, vamos construir um modelo equivalente no ARENA. Nossos blocos básicos ficam como:



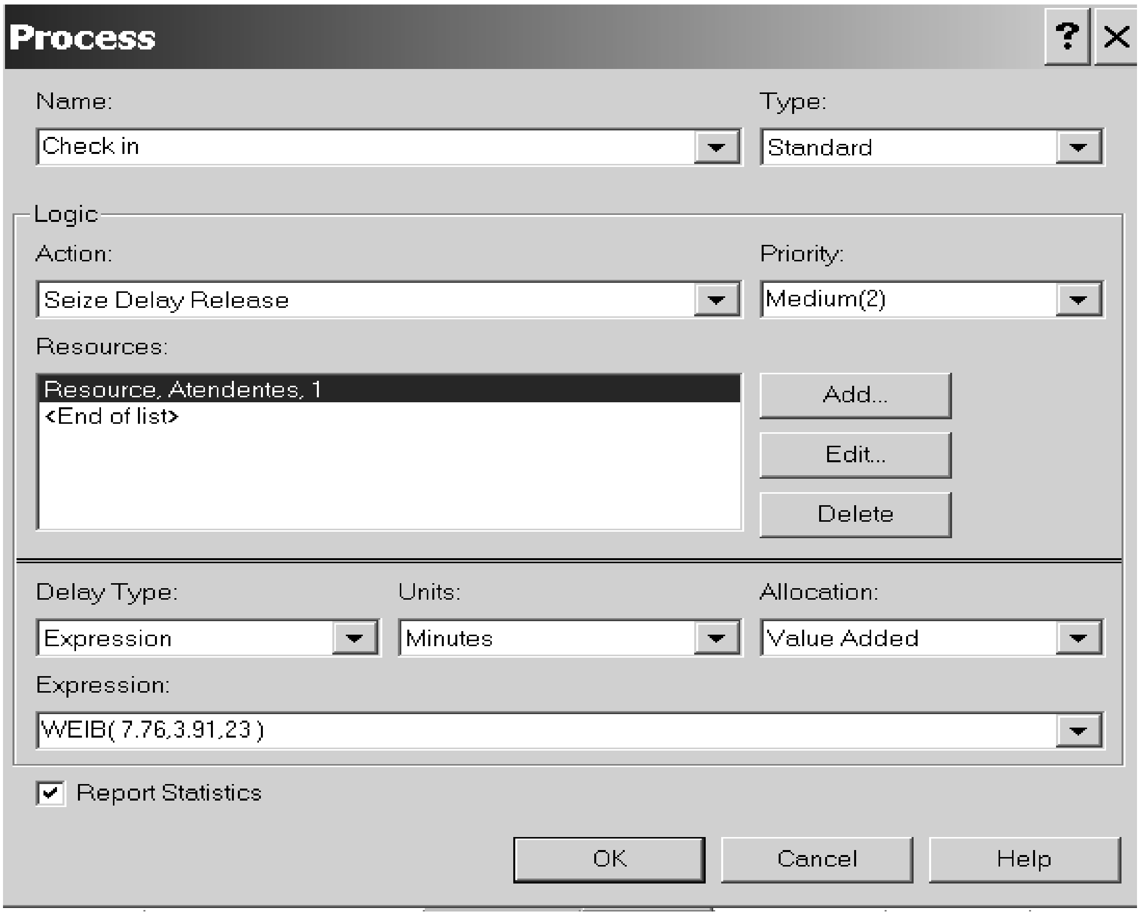
O bloco de chegada é idêntico ao que já fizemos no exemplo anterior e escolhemos EXPO(1.6,7) para a distribuição exponencial, usando a série 7 da RAND4. Vamos chamar de Passageiros a entidade que vai circular na simulação. Temos então:



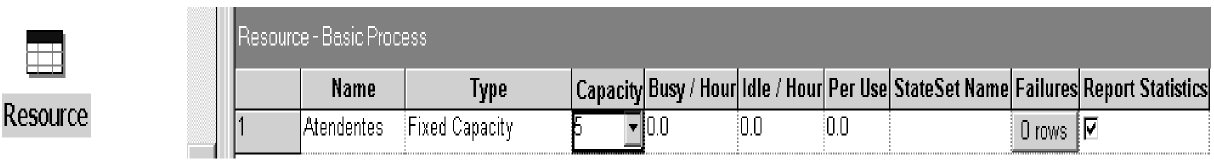
O 2º bloco é referente a procura pelo balcão de check-in. A ação, neste caso é só *Delay* porque só há um atraso, uniformemente distribuído, para chegar até o balcão. O quadro preenchido fica da seguinte forma:



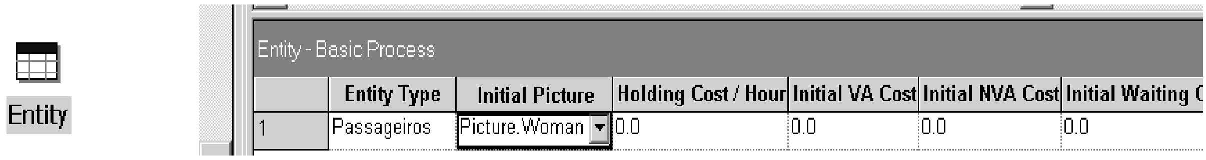
O bloco do check-in também é bastante similar ao que vimos anteriormente com a diferença de que escolhemos o nome de Atendentes para o recurso e a distribuição do atendimento é escolhida como WEIB(7.76,3.91,23), ou seja uma Weibull com os parâmetros dados e usando-se a série 23 da RAND4. Temos então:



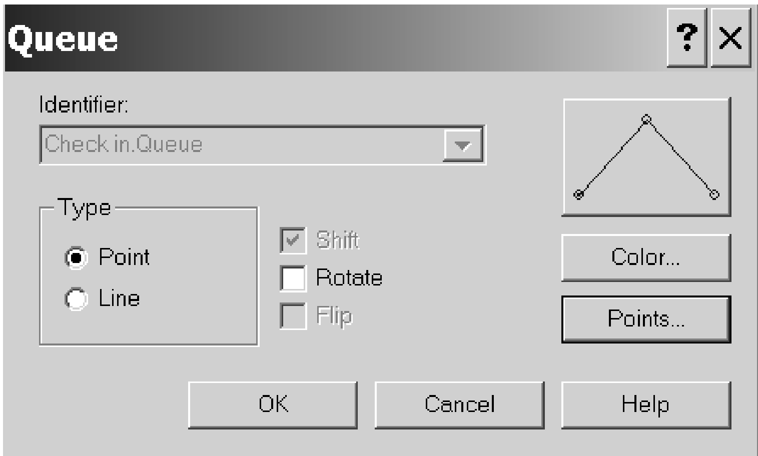
No bloco *Dispose*, muda-se apenas o nome para Saída para o Embarque.
Neste momento devemos nos lembrar que temos 5 atendentes no balcão de check-in. Como informar isto ao modelo ? No lado esquerdo da tela do ARENA (onde estão os blocos), rolando com o mouse vão aparecer desenhos de planilhas e uma delas é referente a *Resources*. Clicando nela vai aparecer, na parte inferior, uma planilha referente aos recursos (atendentes no nosso caso). Um dos campos da planilha é *Capacity* que deverá estar com 1. Deve ser alterado para 5, como mostrado a seguir:



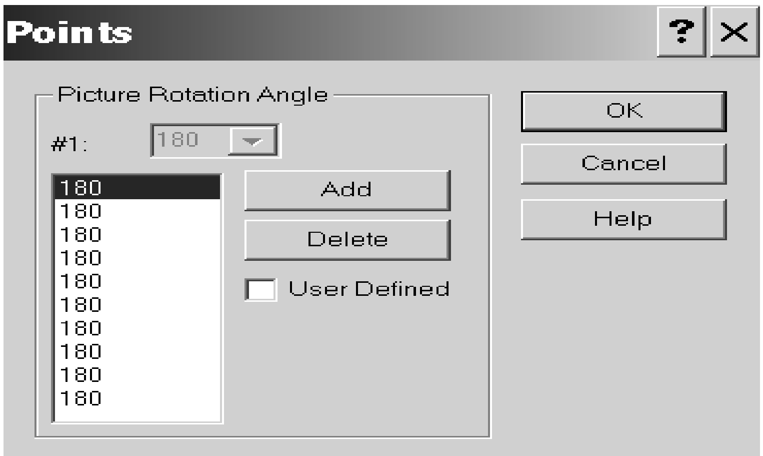
Podemos agora escolher um símbolo que represente os passageiros na dinâmica da simulação. No lado esquerdo, clicamos a planilha *Entity*. Vai abrir, em baixo a direita, a planilha relativa aos passageiros. No campo *Initial Picture*, escolhemos a figura *Picture.women* que é um boneco representando uma mulher, como podemos ver a seguir:



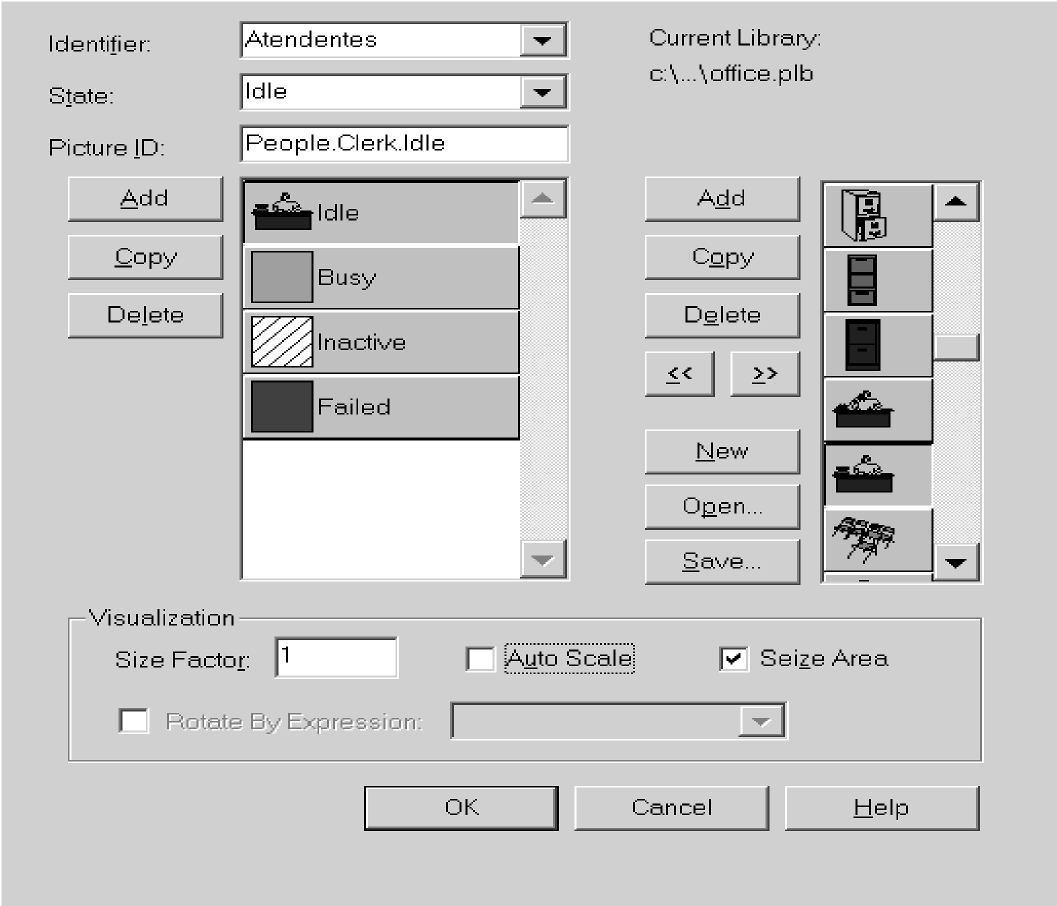
Vamos alterar a forma de representação da fila de linha para pontos. Dando um duplo click na linha, o seguinte quadro aparece:



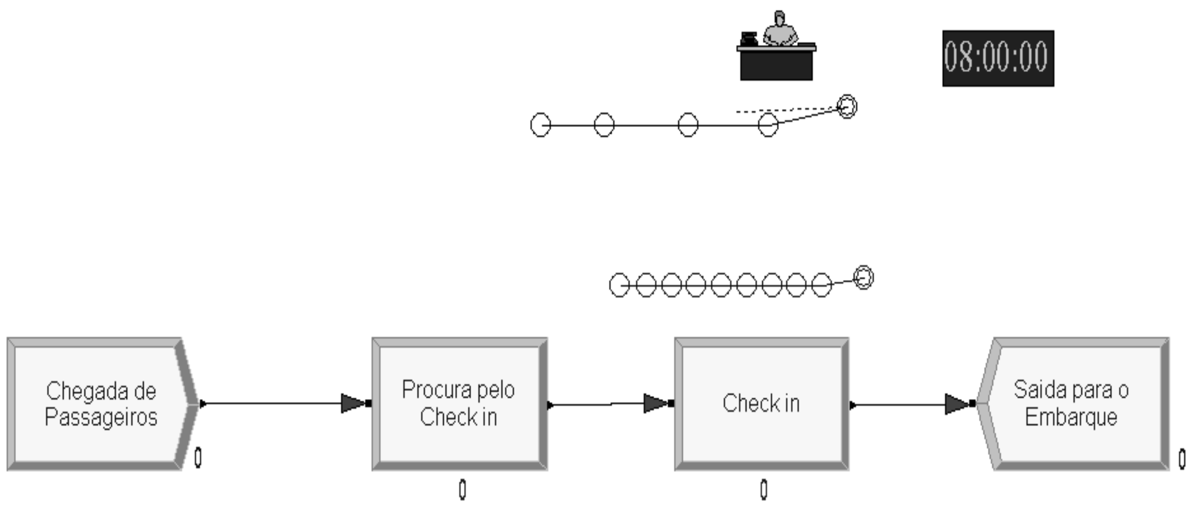
Mudamos de *Line* para *Point* e depois clicando em *Points..*, vai abrir outro quadro onde, clicando no botão *Add*, podemos ir colocando pontos ou seja lugares na fila. Neste exemplo colocamos 10 pontos. No final clica-se em **OK**.



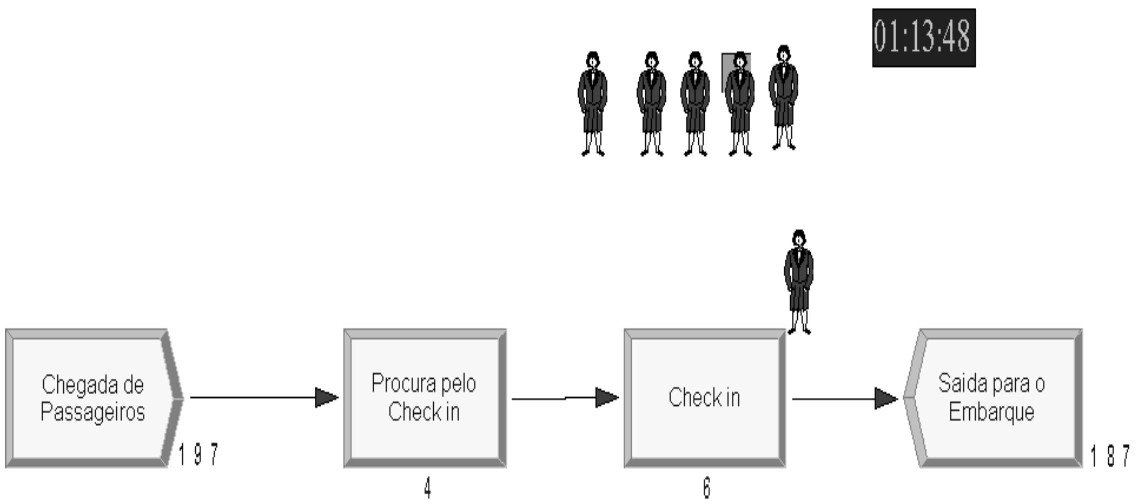
Para dar um toque mais realista a simulação, vamos acrescentar o balcão do check-in e os 5 pontos de atendimento. Iniciamos clicando, na 2ª barra de botões em cima, em  que fará com que a seguinte tela apareça:



Escolhendo Atendentes como *Identifier*, podemos selecionar figuras para representar os atendentes ociosos (*Idle*) ou ocupados (*Busy*). É importante notar que as figuras que aparecem fazem parte da chamada biblioteca padrão. Podemos (clicando no *open* deste quadro) abrir (*Open*) outras bibliotecas que possuem outras figuras.
Para escolher uma figura para *Idle*, por exemplo, clica-se em *Idle*, clica-se na figura desejada e depois clica-se em «. A figura que escolhemos, está na biblioteca *Office* do próprio Arena.
Antes de clicar em **OK**, devemos marcar *Seize Area*. Isto faz com que junto a figura seja criada uma linha (semelhante a da fila) para identificar os pontos de atendimento. No nosso exemplo criamos 5 pontos (de maneira similar ao feito para a fila) pois temos 5 atendentes.
Para finalizar, vamos colocar um relógio para melhor visualização do andamento da simulação. Para tanto basta clicar no botão  (clock) e arrastá-lo apara a área de trabalho. Escolhemos 8:00 horas como a hora inicial marcada pelo relógio.
Nosso modelo está pronto e sua aparência é:



Ajustamos a seguir os parâmetros da simulação (**Run** → **Setup**) e escolhemos 14 horas para a duração da simulação (das 8:00 às 22:00 horas). A figura a seguir mostra um retrato da simulação:



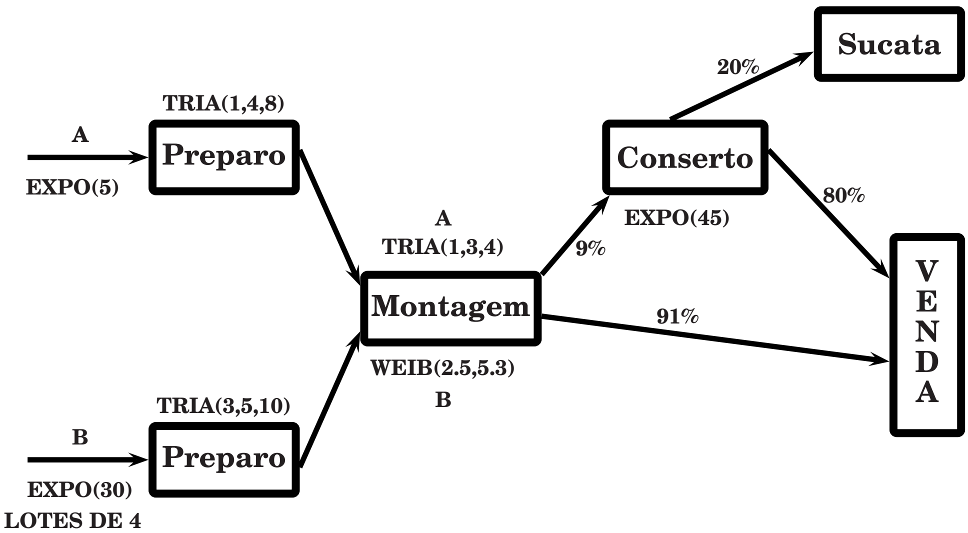
O relógio marca 01:13:48, ou seja já temos mais de 5 horas de simulação sendo que 197 passageiros já chegaram, 187 já passaram pelo check-in sendo que os 5 atendentes estão ocupados e 1 passageiro aguarda na fila. Temos também 4 passageiros procurando pelo balcão de check-in.
Dos relatórios emitidos ao final da simulação, podemos extrair, entre outros, os seguintes dados:
Passageiros atendidos pelo check-in das 8:00 às 22:00 : 524
Tempo total gasto, em média, por um passageiro, da entrada a saída : 13,19 minutos
Menor tempo gasto por um passageiro, da entrada a saída : 4,91 minutos
Maior tempo gasto por um passageiro, da entrada a saída : 29,35 minutos

Percentual de ocupação dos Atendentes : 89% do tempo
Número médio de clientes na fila do check-in : 2,26 passageiros
Maior tempo gasto por um passageiro na fila do check-in : 26,90 minutos

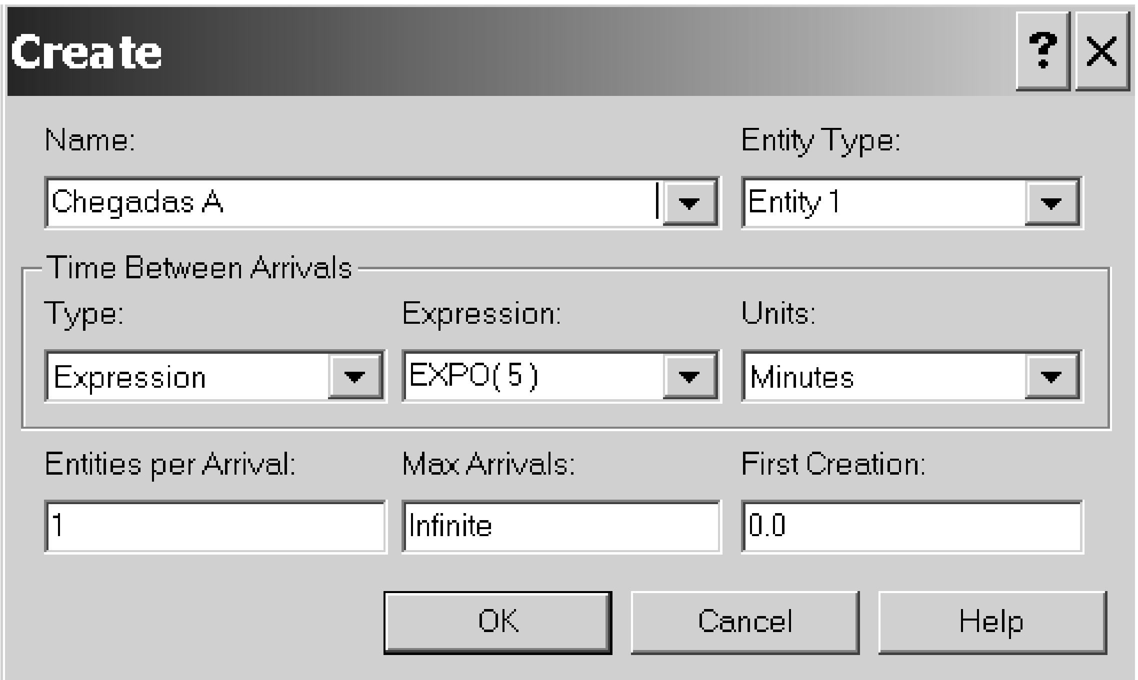
Como já citado, dezenas de outros relatórios podem ser impressos com dados da simulação.

5.3.8 Simulação de um processo de produção

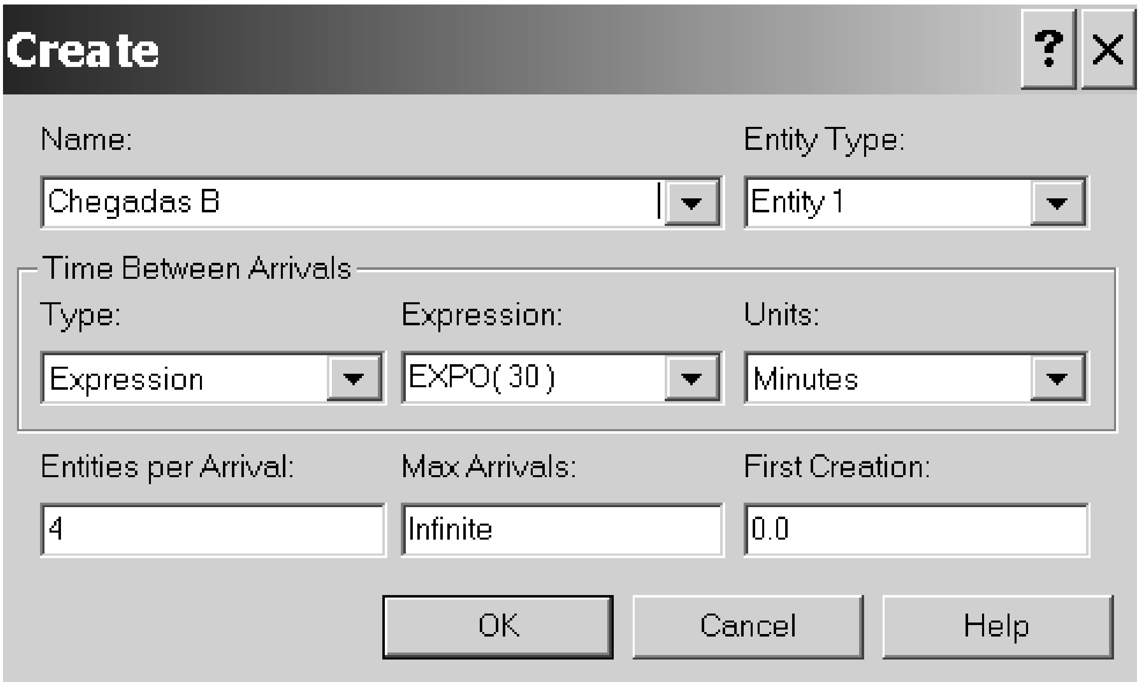
Uma indústria eletrônica produz 2 medidores eletrônicos que chamaremos de **A** e **B**. Basicamente os 2 aparelhos, embora com funções distintas, tem as mesmas características quais sejam, uma placa de circuito impresso, onde são soldados componentes, e uma embalagem de plástico com alguns botões.
As placas de circuito impresso e as embalagens são produzidas externamente e, ao chegar a indústria, são enviadas para uma área onde, as placas, são "preparadas" para receber os componentes que serão soldados. Por sua vez, as embalagens são pintadas e colocados os botões. Estas áreas de preparação são, dadas as características técnicas, diferentes para os aparelhos **A** e **B**.
Após a preparação, as placas e embalagens são enviadas para a área de montagem (única para **A** e **B**) onde os componentes são soldados e o aparelho é montado. Nesta área é, ao final, feita uma inspeção de qualidade onde dados históricos tem mostrado que 91% dos aparelhos acabados estão perfeitos e são enviados diretamente para a seção de venda. Os que não passam no controle de qualidade (9%), são enviados para um outro setor onde as causas da rejeição são analisadas e, se for o caso, corrigidas.
Dados estatísticos mostram que 80% dos aparelhos que caem neste setor conseguem ser recuperados e vendidos enquanto que os restantes 20% vão para sucata.
Foram efetuadas amostras estatísticas em todo o processo e ficou comprovado que todas as atividades (chegadas, preparação, montagem, etc...) são processos aleatórios que seguem determinadas distribuições. A figura a seguir dá a visão geral de todo o processo:



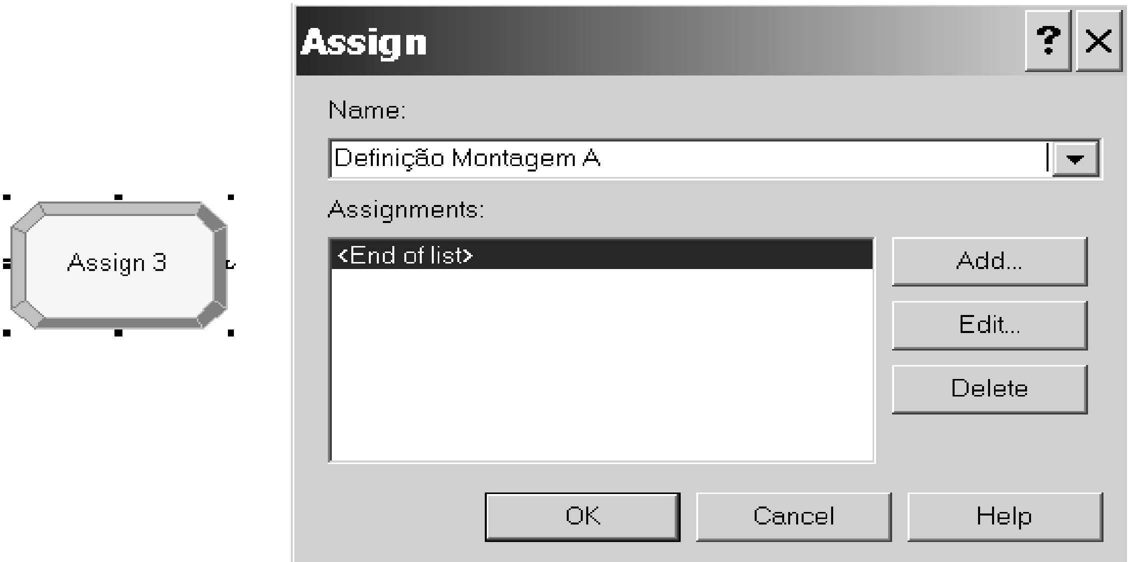
As placas e embalagens do produto **A**, chegam seguindo uma distribuição exponencial com intervalo médio de 5 minutos entre chegadas. Já as placas e embalagens do modelo **B** chegam, em lotes de 4, também seguindo uma distribuição exponencial com intervalo médio de 30 minutos entre chegadas.
O tempo de preparação de cada unidade tipo **A** segue uma distribuição triangular com parâmetros 1, 4 e 8 minutos enquanto que a do produto **B** também segue uma triangular com parâmetros 3, 5 e 10 minutos.
Após preparados, tanto as placas como embalagens dos 2 produtos vão para a seção de montagem. Lá, cada unidade do produto tipo **A** é montada num tempo que segue uma distribuição triangular com parâmetros 1, 3 e 4 minutos. Já uma unidade do produto **B** é montada em um tempo que segue uma Weibull com $\beta = 2.5$ e $\alpha = 5.3$.
Os produtos rejeitados vão para a seção de "reparo" onde são consertados em um tempo que também segue a distribuição exponencial com duração de 45 minutos.
Deseja-se simular esta situação para determinar algumas varáveis tais como nº de unidade na fila, tempo total de produção, etc...
Vamos construir o modelo no Arena e vamos deixar, neste exemplo, a escolha da série dos números aleatórios por conta do Arena.
Inicialmente vamos construir um bloco "Create" para a chegada dos componentes do produto **A**.



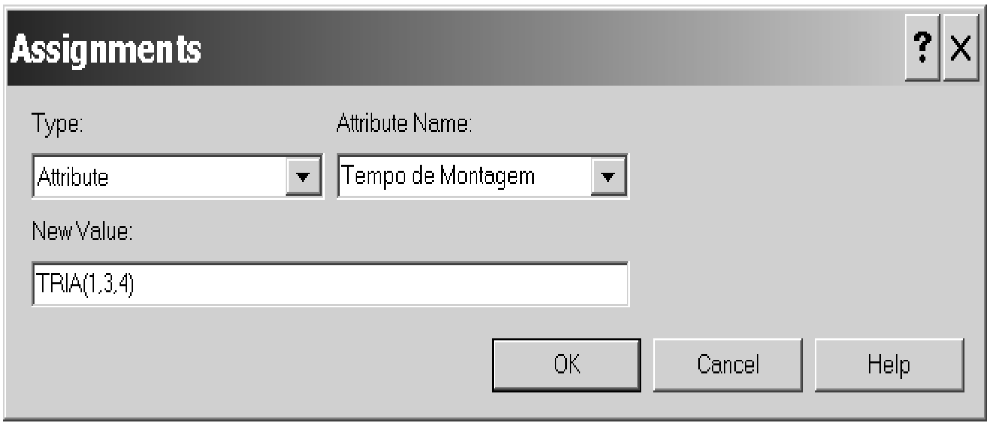
Criamos de forma idêntica um bloco "Create" para as chegadas das placas e embalagens do produto tipo **B**. Temos então:



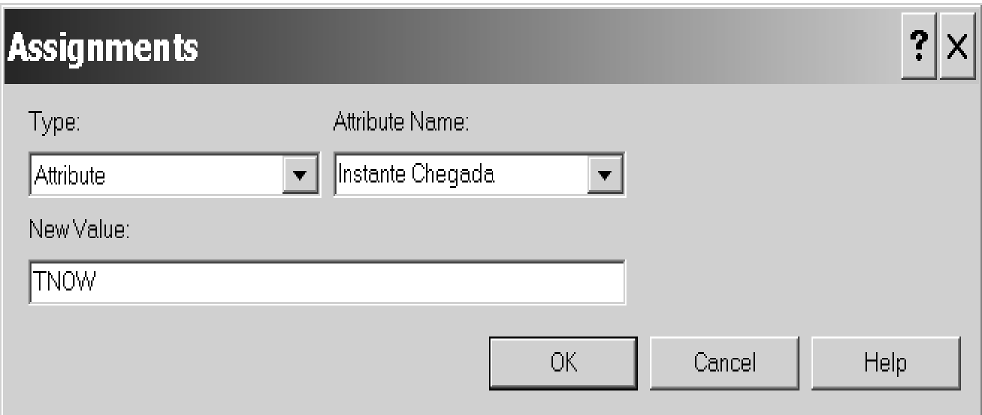
Devemos notar que “Entities per Arrival” foi alterado para **4**, pois as placas e embalagens de **B** chegam em lotes de 4.
Neste ponto temos que nos preocupar com um ponto que não apareceu nos modelos anteriores. Depois de chegar e ser preparado, os componentes do modelo **A** vão para a seção de montagem. No entanto com o modelo tipo **B** ocorre a mesma coisa, ou seja ambos usam a seção de montagem mas com tempos seguindo distribuições diferentes. Não podemos definir 2 distribuições para o mesmo processo nem definir 2 processos porque na verdade só existe uma seção de montagem. Para contornar isto, vamos definir um atributo Tempo de Montagem a quem atribuiremos o tempo de montagem apropriado para cada tipo de produto. O bloco que faz estas atribuições no Arena é o *Assign*. Temos então:



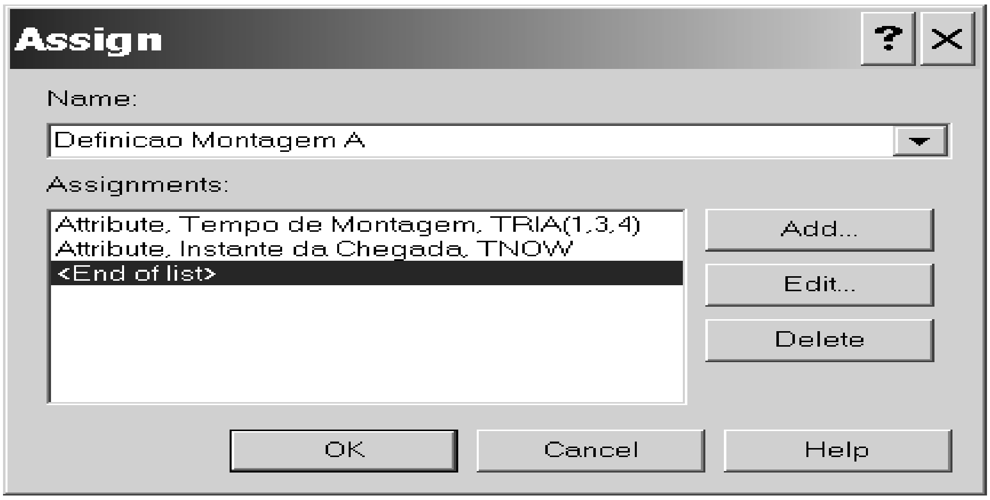
Demos o nome de Definição Montagem A. A seguir, clicamos no botão *Add* e criamos o atributo Tempo de Montagem informando a distribuição do tempo de montagem para o produto tipo A, ou seja, TRIA(1,3,4).



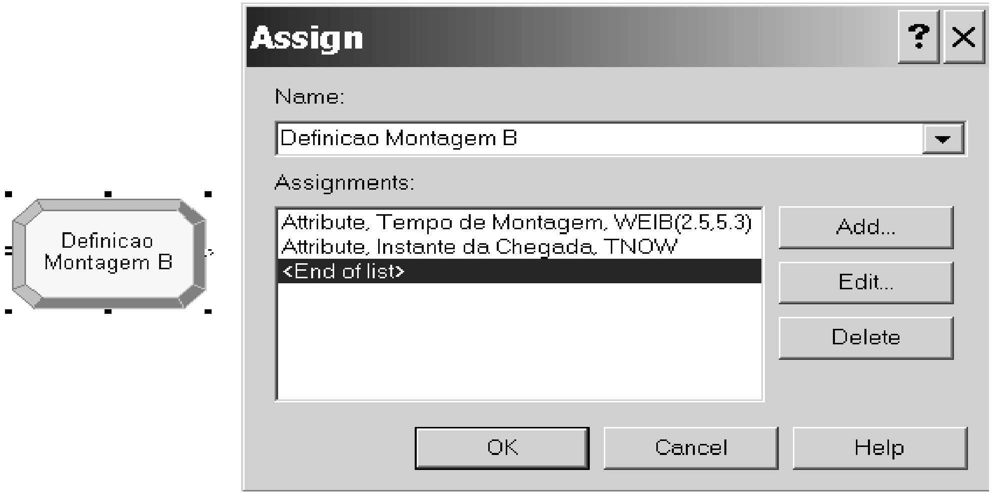
Cliando em *Add* mais uma vez, vamos criar o atributo Instante da Chegada informando o instante da chegada das peças de um modelo **A**. Usamos a variável *TNOW* que é uma variável do Arena que contém o instante do evento que acabou de ocorrer, no caso a chegada dos componentes de um modelo tipo **A**.



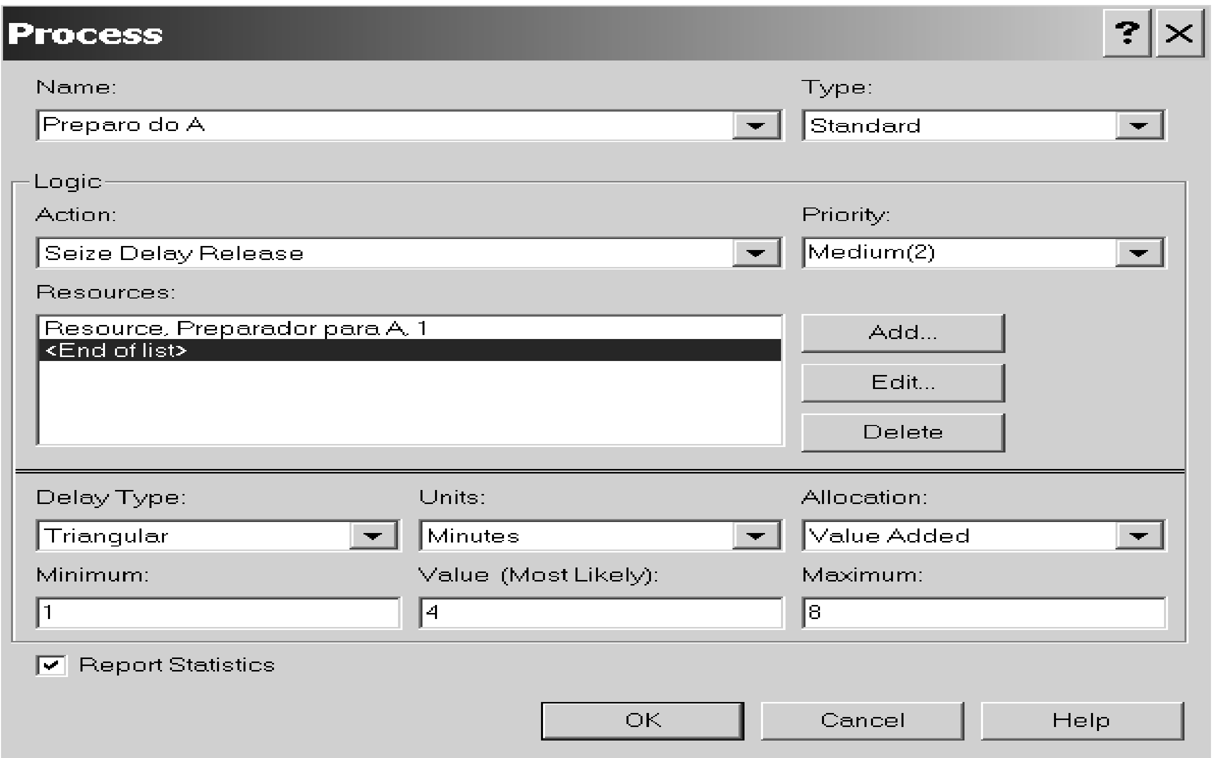
O Tempo de Montagem, para o produto **A** está completo:



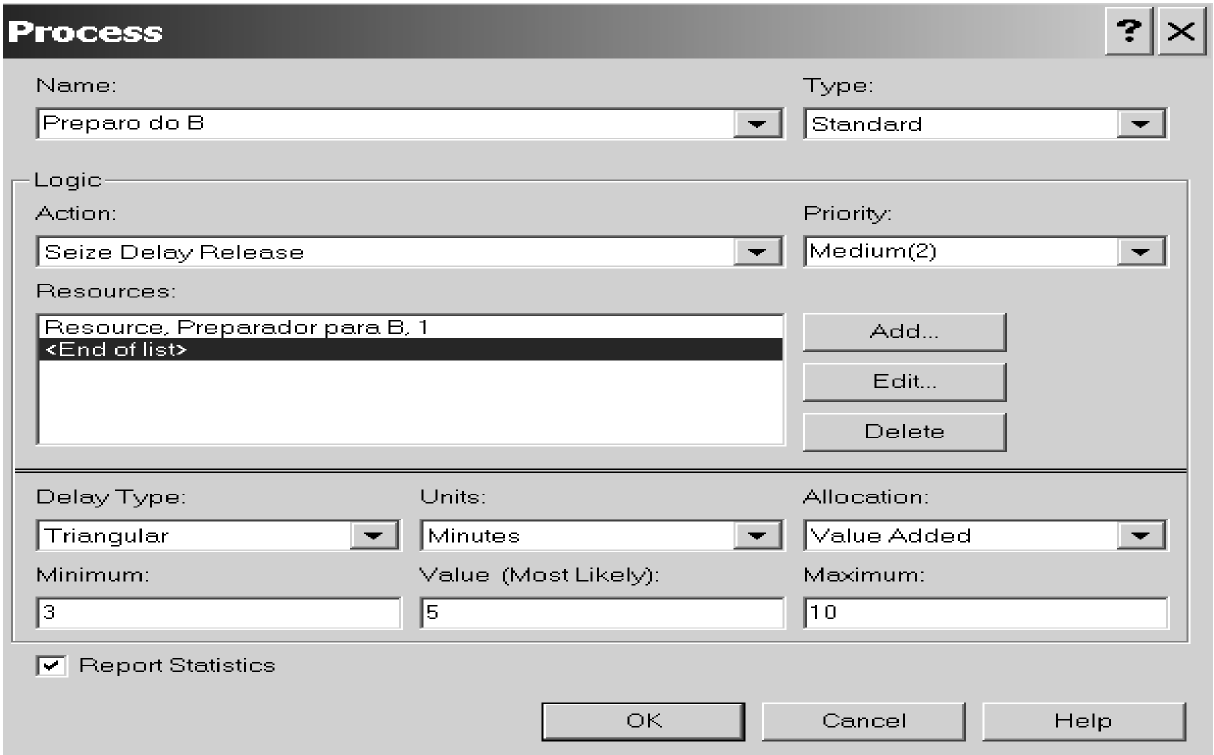
Temos que fazer o mesmo para o produto **B** só tendo o cuidado de escolher os atributos **Tempo de Montagem** e **Instante da Chegada** que tínhamos escolhido anteriormente (eles vão aparecer no *Listbox* dos atributos).



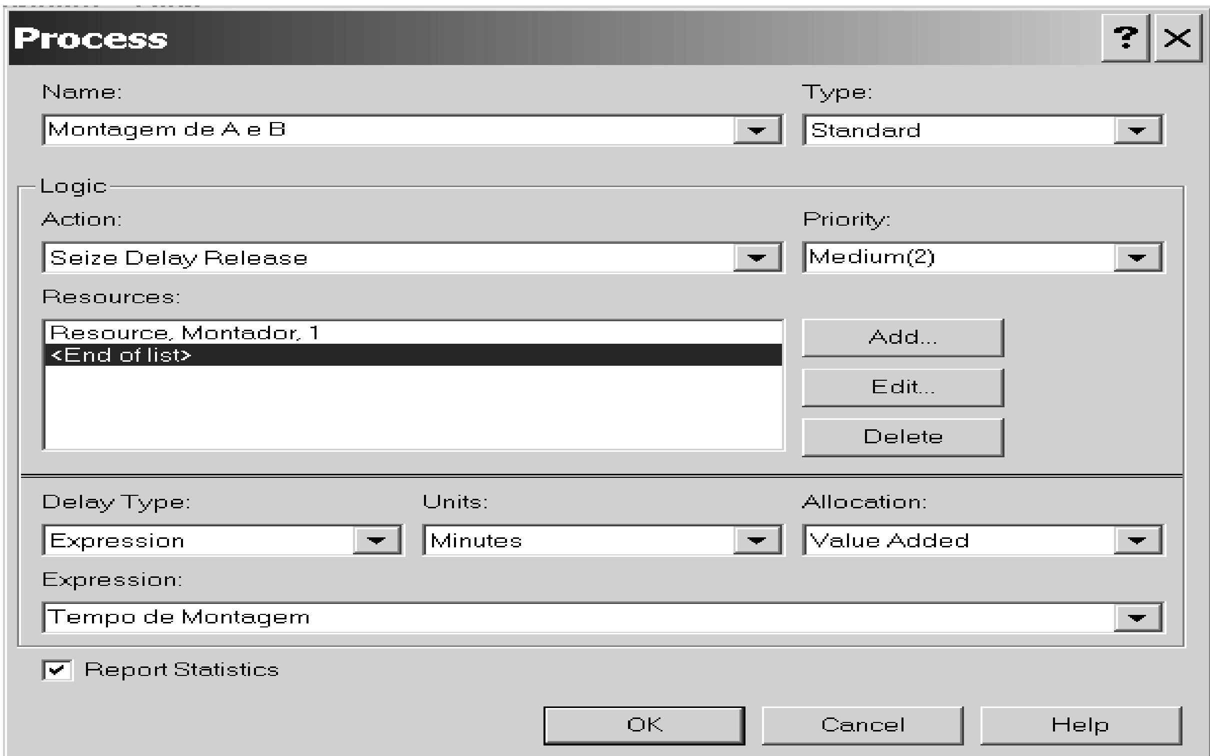
Podemos agora criar os blocos (*Process*) para a etapa de preparação.
Para o tipo **A** temos:



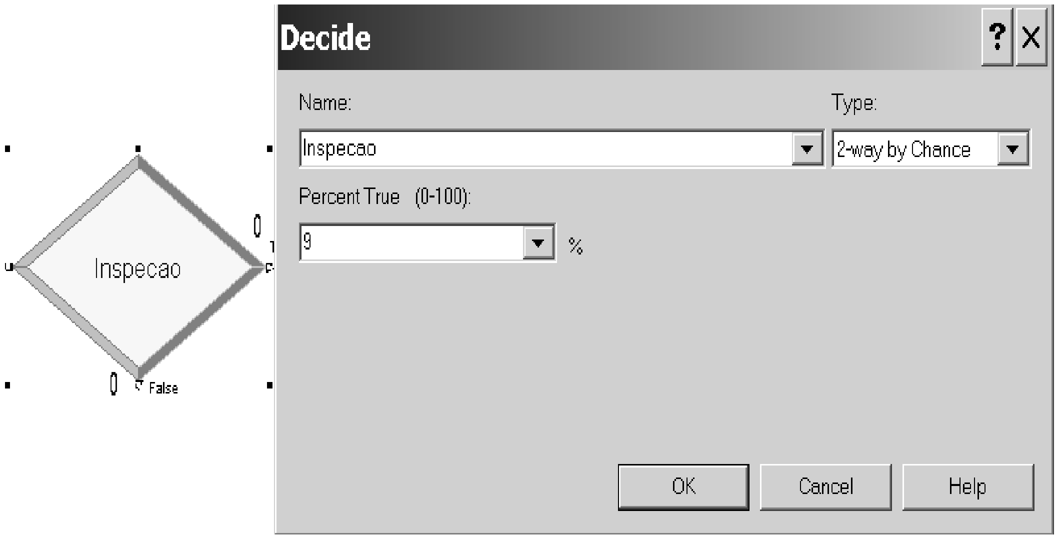
Podemos observar que criamos um recurso que chamamos de "Preparador para A". Para o tipo **B** temos que fazer a mesma coisa:



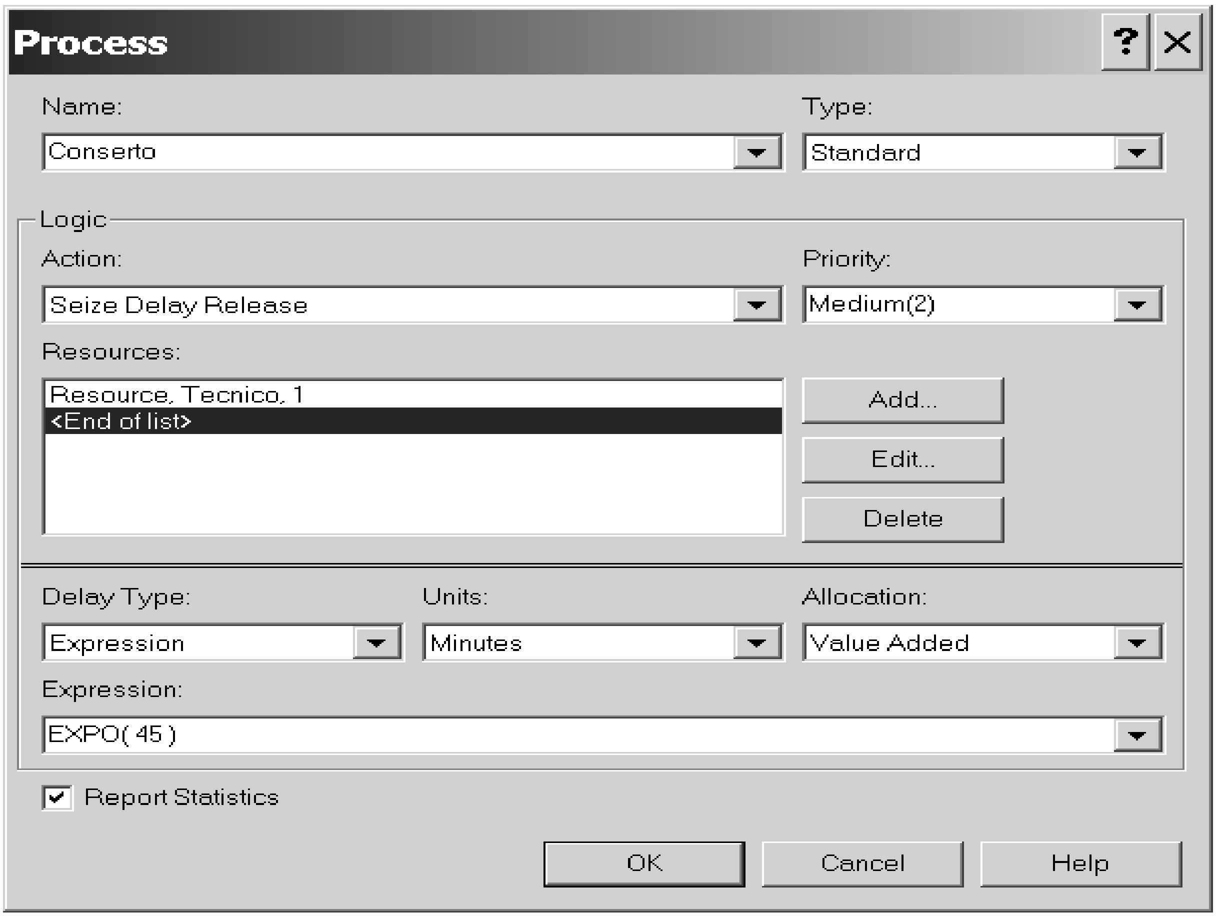
Temos agora a Montagem onde devemos usar o atributo <u>Tempo Montagem</u> que criamos anteriormente.



Após montados, os produtos passam por uma inspeção de qualidade. Podemos representá-la usando o bloco *Decide*. Temos então:

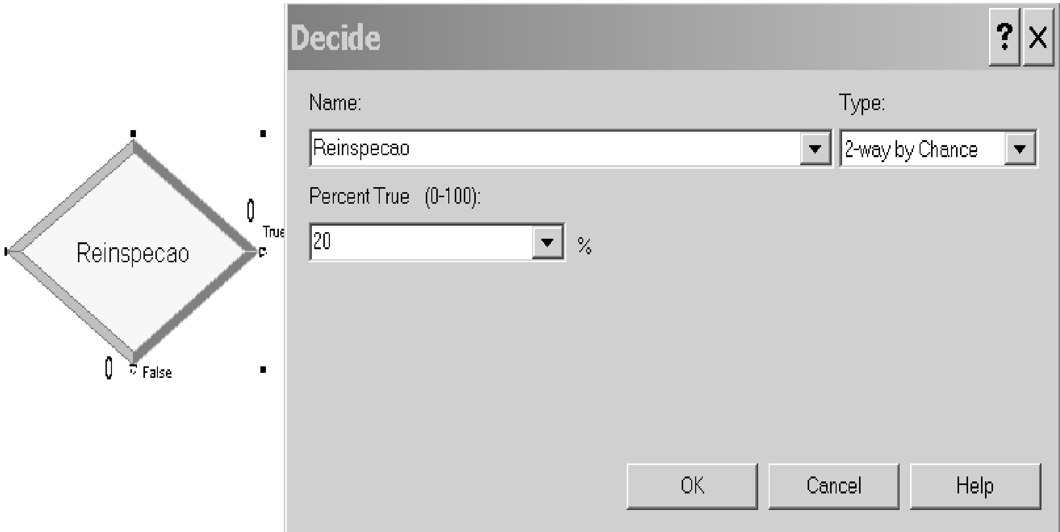


Podemos escolher o percentual, no caso 9% de rejeição. Estes rejeitados vão para a área de conserto que, no Arena, será representado por mais um bloco *Process*.



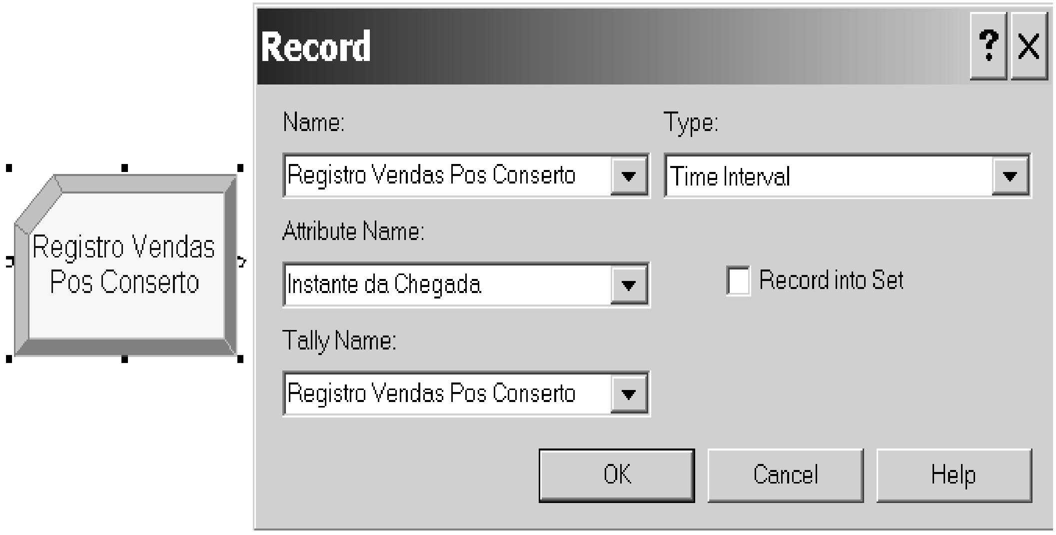
Adicionamos o recurso que vai consertar os produtos rejeitados (Técnico).

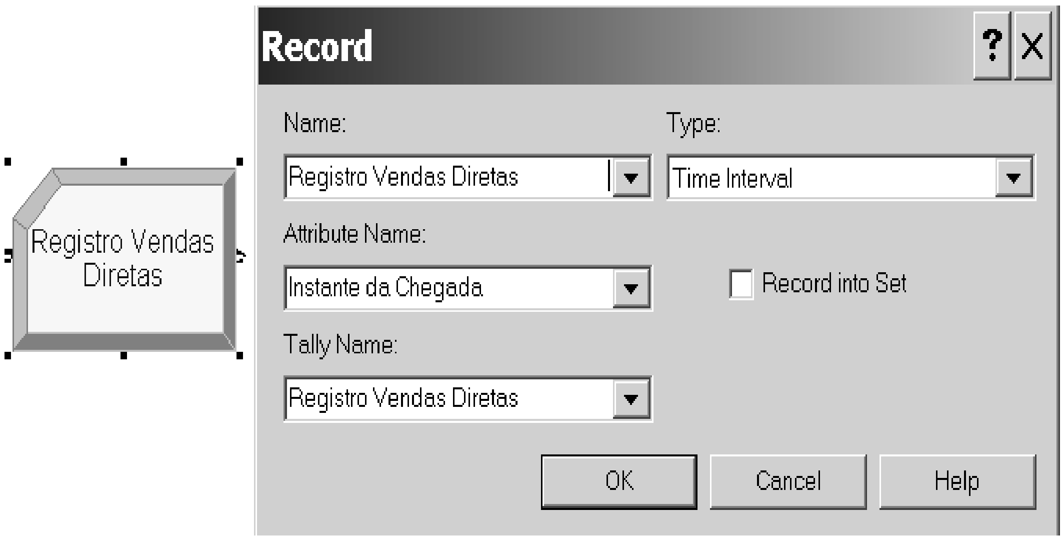
Temos após o conserto, uma re-inspeção para verificar os que podem ser comercializados. Temos então mais um bloco de decisão. Podemos representá-lo usando o bloco *Decide*. Temos então:



Tendo definido todas as operações, precisamos nos preocupar com a obtenção das informações que queremos analisar.Lembre-se que, como parte da própria simulação, nós obtemos estatísticas sobre a utilização dos recursos, tamanho das filas e espera em cada uma das filas. No entanto, queremos também informações sobre a duração do ciclo para os produtos que passam no controle de qualidade e para os que não passam, ou seja, queremos conhecer o ciclo dos que são "consertados" e dos que vão para a sucata. O bloco *Record* permite que se tenha estes dados.
Para os que vão para a sucata, temos:

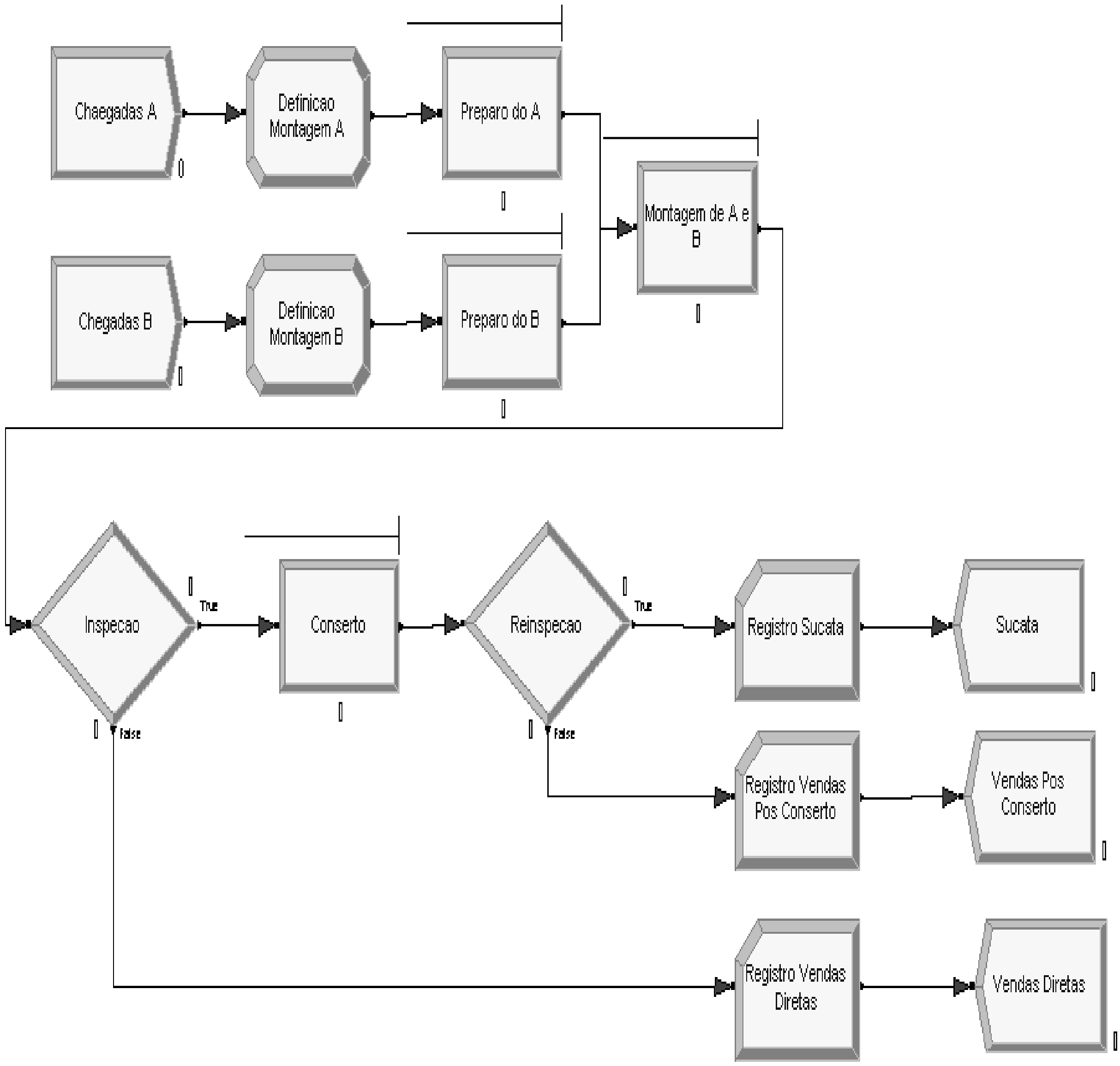
Para *Type*, escolhemos *Time Interval* e no atributo escolhemos Instante da Chegada. Com isto o Arena vai registrar cada duração desde que as partes do produto chegam até irem para a sucata. Para os demais ítens (comercializadas direto e comercializados após reparo), o procedimento é o mesmo como podemos ver a seguir:





Finalmente não podemos esquecer dos módulos de saída, ou seja *Dispose*. Temos 3 "saídas" possíveis: Sucata, Vendas Pós Conserto e Vendas Diretas. Temos que ter 3 blocos *Dispose*.

Em um modelo deste tipo é usual que vá se construindo os "blocos" sem nos preocuparmos com a ligação lógica entre eles. Neste momento, quando todos os blocos já estão prontos, devemos fazer a ligação ente eles de modo a que o Arena possa executar a simulação. Após fazer esta tarefa, nosso modelo tem a seguinte aparência:


Chaegadas A
Definicao Montagem A
Preparo do A
Chegadas B
Definicao Montagem B
Preparo do B
Montagem de A e B
Inspecao
True
False
Conserto
Reinspecao
True
False
Registro Sucata
Sucata
Registro Vendas Pos Conserto
Vendas Pos Conserto
Registro Vendas Diretas
Vendas Diretas

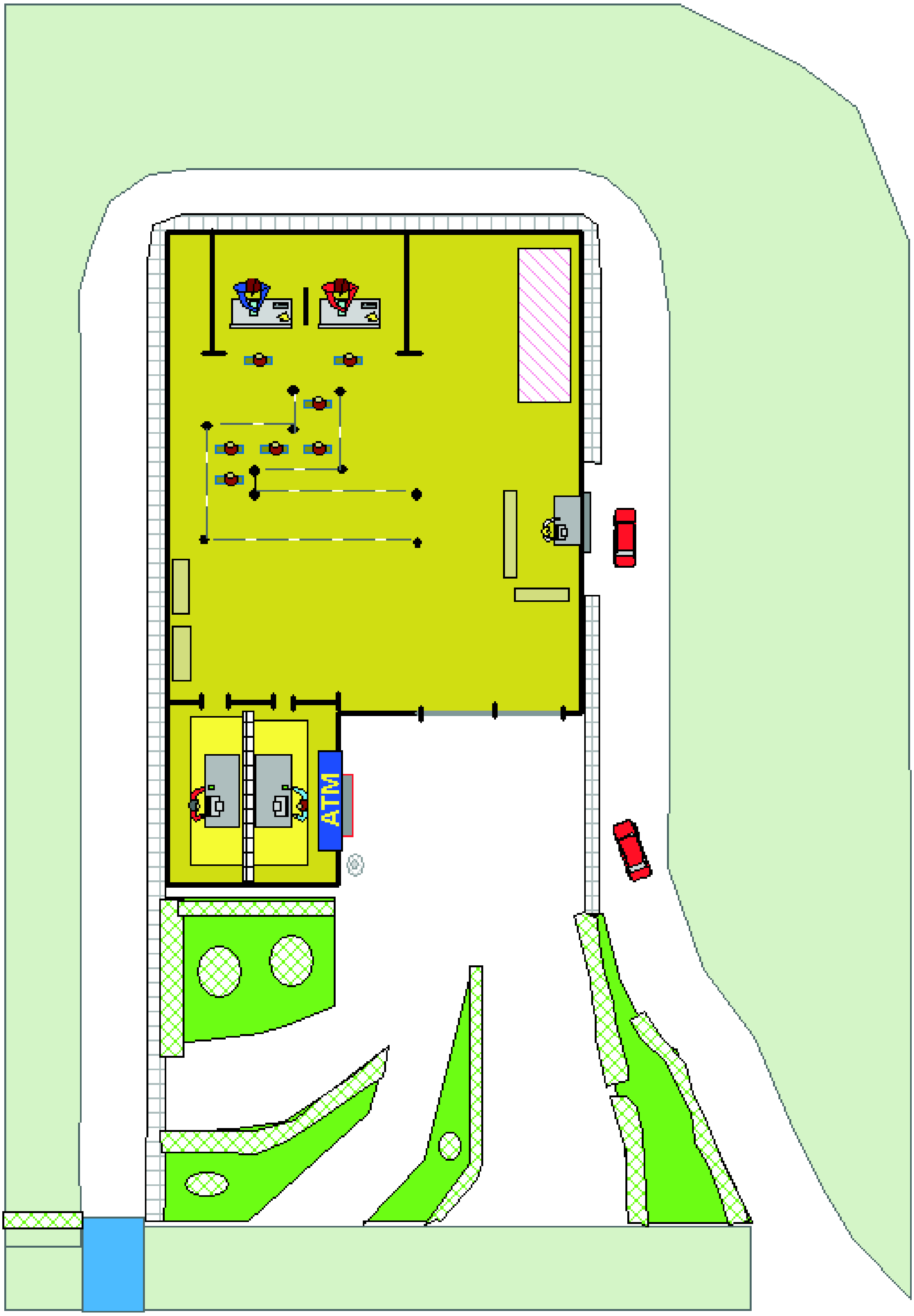
Executando a simulação para um período de 40 horas, os seguintes resultados foram obtidos nos relatórios impressos pelo Arena:

Espera média na Fila (W_q)	**Minutos**
Na prep A	14,98
Na prep B	28,16
Na montagem	2,52
No conserto	536,79
Tamanho médio da Fila (L_q)	
Na prep A	3,18
Na prep B	3,52
Na montagem	0,84
No conserto	15,97
Vendas	**Quantidade**
Direto	728
Consertados	39
Utilização do Recurso (ρ)	%
Prep A	75
Prep B	90
Montagem	85
Conserto	96

Os resultados mostram que temos um problema na área de conserto. Ao final da simulação, existiam 32 modelos na fila aguardando reparo. Pode-se ver também que a área está ocupada quase 100% do tempo. Para se melhorar o processo de produção, teríamos que aumentar a capacidade de produção da seção de conserto.

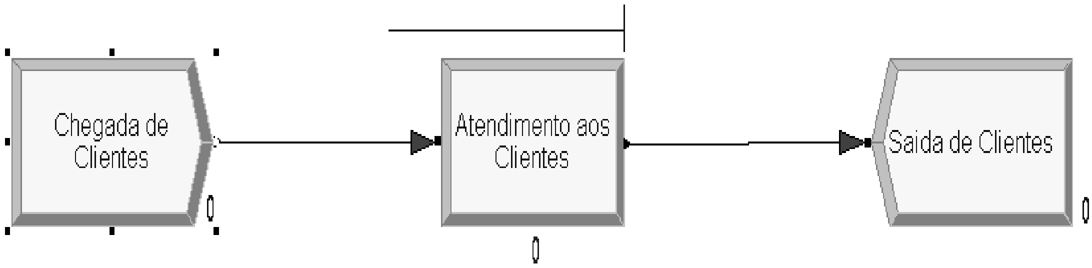
5.3.9 Enfeitando o modelo

O que vimos, até aqui, do ARENA deve representar de 10% a 20% do que é possível fazer com o software. Modelos bastante sofisticados e análise de custos são possíveis de serem realizados. O Arena permite também que modelos de "animação" sejam criados. Um exemplo pode ser visto na página a seguir, onde temos um modelo de uma agência bancária onde além do normal em uma agência, temos caixas de atendimento para clientes motorizados. No entanto, devemos ressaltar que a animação, embora muito interessante como ferramenta de "venda" do modelo, consome tempo para ser construída e, obviamente, os mesmos resultados são obtidos se trabalhamos só com os blocos dos templates.


ATM

5.3.10 Análise dos resultados de saída

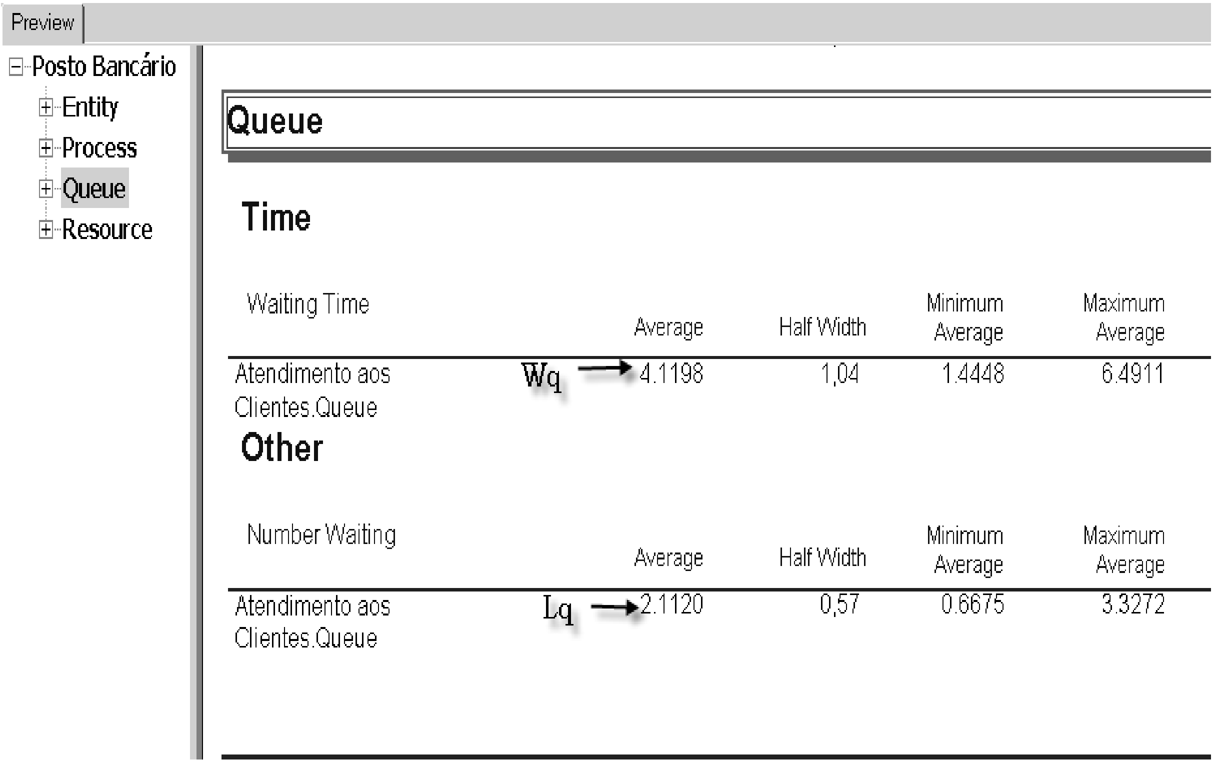
Vamos voltar para o exemplo inicial que vimos para o Arena qual seja o sistema de um pequeno posto bancário com um único caixa (pág. 108).



Vamos executar 10 replicações do modelo cada uma com 1.000 minutos de duração. Na nossa análise, vamos examinar 2 variáveis: o tempo médio que um usuário permanece na fila (W_q) e o número médio de usuários na fila (L_q).
Para as 10 replicações, os resultados encontrados para estas variáveis foram:

Replicação	W_q	L_q
1	3.51	1.72
2	6.49	3.14
3	3.70	1.91
4	4.88	2.62
5	4.30	2.34
6	6.03	3.33
7	3.56	1.70
8	4.23	2.22
9	3.06	1.47
10	1.44	0.67

Examinemos os valores de W_q. Qual o valor correto ? 1.44 ou 6.49 ?
Não podemos esquecer que estamos lidando com eventos aleatórios seguindo determinada distribuição (a exponencial no exemplo) e a grande variabilidade dos resultados encontrados [1.44 – 6.49] é, na verdade, um resultado esperado.
A média do W_q para as 10 replicações é 4.12 minutos com desvio padrão igual a 1.46 minutos.
A média pode ser vista no próprio Arena no relatório *Category Overview*. Este relatório dá o resumo das n replicações que efetuamos (10 no nosso exemplo).



Preview

- Posto Bancário
  - Entity
  - Process
  - Queue
  - Resource

Queue

Time

Waiting Time		Average	Half Width	Minimum Average	Maximum Average
Atendimento aos Clientes.Queue	Wq →	4.1198	1,04	1.4448	6.4911

Other

Number Waiting		Average	Half Width	Minimum Average	Maximum Average
Atendimento aos Clientes.Queue	Lq →	2.1120	0,57	0.6675	3.3272

Embora não seja dado o desvio padrão, podemos ver um outro valor: *Half Width* que é a metade do intervalo de confiança da média. O intervalo de confiança é a faixa de valores em que acredita-se que esteja a média, com uma determinada probabilidade $(1-\alpha)$.

O valor do meio intervalo (h) é dado por:

$$h = \frac{t \times s}{\sqrt{n}}$$

onde:

$t \Rightarrow$ é o $(1-\alpha/2)$ percentual da distribuição t de *Student* com $n-1$ graus de liberdade.

$s \Rightarrow$ é o desvio padrão da amostra e,

$n \Rightarrow$ é o tamanho da amostra.

O Arena usa um valor de α igual a 5% como *default*.

Para o W_q do nosso exemplo, temos:

$n = 10$ replicações.

Na tabela de t (pág. 149), para $\nu = n-1 = 9$ graus de liberdade e para $(1-\alpha/2) = 0.975$ temos $t = 2.26$ (algumas tabelas de t são por $\alpha/2$, ou seja, 0.025).

Como já vimos, $s = 1.46$.

O valor do meio intervalo é igual a:

$$h = \frac{2.26 \times 1.46}{\sqrt{10}} = 1.04, \text{ como pode ser visto no relatório acima.}$$

Após 10 replicações do nosso modelo, o que o ARENA nos contou para o valor de W_q, é que, com 95% de probabilidade, o valor esperado do tempo de espera de um cliente na fila, está dentro do intervalo $[4.12 \pm 1.04]$ ou $[3.08\,;\,5.16]$.

E se desejarmos uma faixa que nos dê 99% de probabilidade de conter a média ?
Neste caso o valor de t para $\alpha = 1\%$ com $\nu = 9$ graus de liberdade é 2.82 (pág. 149) ficando o valor do meio intervalo como:

$$h = \frac{2.82 \times 1.46}{\sqrt{10}} = 1.3$$

O intervalo de confiança, com 99% de probabilidade de conter a média, passa a ser $[4.12 \pm 1.3]$ ou $[2.82\,;\,5.42]$
Como aumentamos a confiabilidade, temos uma faixa mais larga.

Se quisermos um meio intervalo de 0.5, por exemplo, quantas replicações devemos fazer ?
Vamos supor que para um α igual a 5%, ou seja com 95% de certeza, queremos que a faixa do W_q esteja no intervalo $\bar{x} \pm 0.5$.
Para encontrar o número de replicações (n) necessárias para que o meio intervalo fique em torno de 0.5, podemos usar a seguinte aproximação (Kelton [2]):

$$n = n_0 \frac{h_0^2}{h^2}, \quad \text{onde:}$$

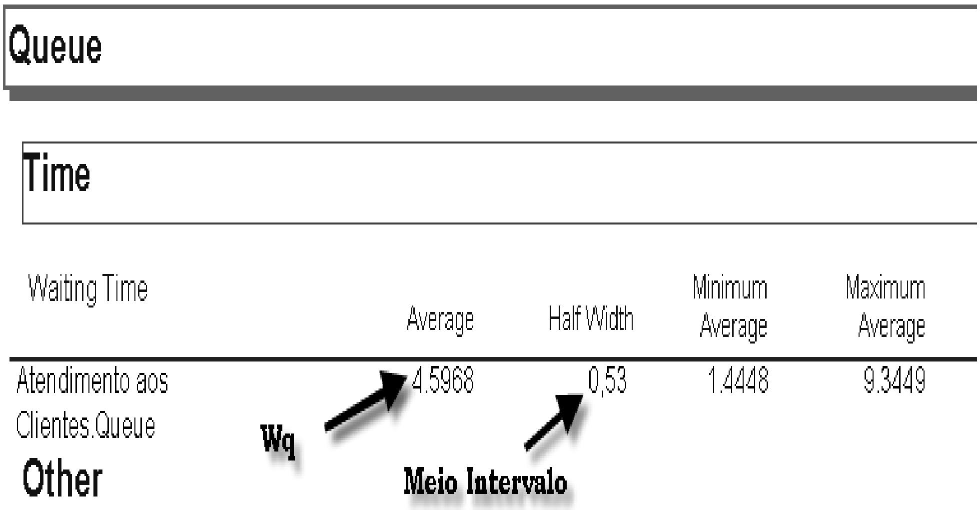
$n \Rightarrow$ número de replicações a serem executadas
$n_0 \Rightarrow$ número de replicações da execução “inicial”. Geralmente um valor entre 10 e 20.
$h_0 \Rightarrow$ valor do meio intervalo encontrado na execução inicial.
$h \Rightarrow$ meio intervalo desejado.

No nosso exemplo como $n = 10$, $h_0 = 1.04$ e queremos $h = 0.5$, temos:

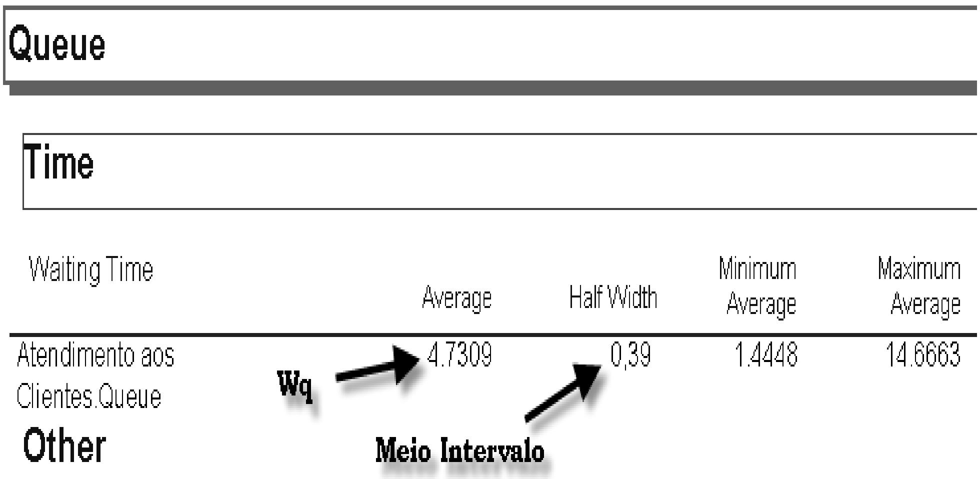
$$n = 10 \frac{(1.04)^2}{(0.5)^2}$$

$$n = 43.26 \simeq 44 \text{ replicações}$$

Executando o ARENA, alterando-se o modelo para que sejam feitas 44 replicações, obtemos o seguinte resultado:



Como podemos observar, o W_q, média das 44 replicações, é igual a 4.60 minutos e o meio intervalo é igual a 0.53 (perto do objetivo, 0.50).
Sendo assim, podemos afirmar, com 95% de certeza, que o tempo médio que um cliente fica na fila do posto bancário (W_q), está no intervalo 4.60 ± 0.53, ou seja, $[4.07\,;\ 5.13]$.
Com o aumento do número de replicações, a faixa em que o W_q está, se reduz significativamente e poderíamos inferir que aumentando-se a número de replicações teremos resultados mais precisos. O resultado abaixo é o conseguido após 100 replicações:



Com 95% de certeza, podemos afirmar que W_q está no intervalo $[4.34\,;\ 5.12]$.
Para 200 replicações, o intervalo passa a ser $[4.23\,;\ 4.73]$.
Poderíamos aumentar o número de replicações (com infinitas replicações, teríamos precisão total) mas não podemos esquecer que cada replicação adicional faz com que o tempo de processamento seja maior.
O nosso exemplo é, provavelmente, o modelo mais simples que pode ser construído.

Mesmo assim, a execução das 200 replicações, em um micro bastante rápido, levou mais de 15 minutos. Para um modelo do mundo real e com razoável complexidade, poderia levar horas ou mesmo dias.
Assim sendo, o cotejamento entre a obtenção dos resultados com determinada precisão e o tempo de processamento necessário para isso, deve estar sempre em avaliação pelo analista que conduz o modelo de simulação.

5.4 Exercícios

1) Usando o método dos quadrados médios, calcule os 12 primeiros números gerados, com 4 dígitos, a partir de uma semente igual a 7308.

2) Use o método congruente linear para gerar um seqüencia de 3 números aleatórios de 2 dígitos. Use $x_0 = 27$, $a = 8$, $c = 47$ e $m = 100$.

3) Encontramos algum problema no exercício anterior se $x_0 = 0$?

4) Considere o método congruente multiplicativo para os seguintes casos:

   a) $a = 11 \quad m = 16 \quad x_0 = 7$

   b) $a = 11 \quad m = 16 \quad x_0 = 8$

   c) $a = 7 \quad m = 16 \quad x_0 = 7$

   d) $a = 7 \quad m = 16 \quad x_0 = 8$

   Gere, para cada caso, todo o período. O que podemos inferir dos resultados encontrados ?

5) Construa, para a RAND2, um cubo idêntico ao mostrado na página 19 para o gerador RANDU.

6) Pesquise na internet para descobrir qual a fórmula usada pela planilha *Excel* no seu gerador de números aleatórios padrão.

7) Desenvolva um gerador de números aleatórios, usando o método da transformação inversa, para a seguinte distribuição probabilística:

$$f(x) = \begin{cases} e^{2x}, & -\infty < x \leq 0 \\ e^{-2x}, & 0 < x < \infty \end{cases}$$

8) Idem para:

$$f(x) = \begin{cases} \dfrac{1}{3}, & 0 \leq x \leq 2 \\ \dfrac{1}{24}, & 2 < x \leq 10 \\ 0, & x > 10 \end{cases}$$

9) Com uma calculadora que tenha a função para gerar números aleatórios uniformemente distribuídos em $[0, 1]$, gere 10 números seguindo a distribuição de Poisson com média (λ) igual a $2,5$.

10) Modifique o programa da página 50 para calcular a probabilidade de um valor (x) de uma distribuição normal, com média (μ) igual a 1.500 e desvio padrão (σ) igual a 300, estar entre 1.300 e 1.800. Use, no programa, a fórmula da distribuição normal:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(X-\mu)^2}{2\sigma^2}}$$

Compare o valor obtido pela simulação com o valor obtido da tabela normal.

11) A gerente de uma loja de eletro-domésticos está desconfiada que o seu estoque de fogões está acima do que seria necessário. Antes de modificar a política de estoques, ela registrou o número de fogões vendidos, diariamente, nos últimos 25 dias. Os dados encontrados estão mostrados a seguir:

Fogões vendidos	2	3	4	5	6
Número de dias	4	7	8	5	1

a) Use os dados para estimar a distribuição de probabilidade das vendas diárias de fogões.

b) Calcule a média da distribuição obtida na parte (a).

c) Descreva como números aleatórios uniformemente distribuídos em $[0, 1]$ podem ser usados para simular as vendas diárias.

d) Usando os números aleatórios 0.4475, 0.9713 e 0.0629, simule as vendas diárias de 3 dias.

e) Usando uma planilha eletrônica (*Excel*, por exemplo), faça um modelo para simular as vendas diárias. Realize 300 replicações e obtenha a média de vendas diárias.

12) Utilizando como sementes os valores 1234, 76545, 88787, 77712, 564783 e 5434 encontre os 3 primeiros números gerados pela RAND4.

13) Pesquise e descubra os comandos para a geração de números aleatórios no Turbo Pascal, inclusive com a escolha da semente.

14) Utilizando uma planilha eletrônica (*Excel*, por exemplo), construa um modelo para a seguinte situação: Um posto de gasolina do governo, que tem somente uma bomba de gasolina, está sempre aberto e tem 2 tipos de clientes. Uma ambulância chega, exatamente, a cada 30 minutos, com o 1º carro chegando no "instante", 15 minutos. Carros de outras repartições públicas, que não são ambulâncias, chegam com um intervalo médio entre chegadas, exponencial, de $5,6$ minutos, com o 1º carro chegando no instante 0. O tempo de serviço, para todos os tipos de carros, tem uma média de $4,8$ minutos (exponencial).
Um carro que chega e encontra a bomba vazia vai ser atendido imediatamente enquanto que os que chegam com a bomba ocupada, formam uma fila única. Isto só não vale para as ambulâncias que, ao chegar, vão imediatamente para o

início da fila (assuma que, se já tem uma ou mais ambulâncias no início da fila, esta nova chegada passa a ser a 1ª da fila). Considere que no início da simulação (instante 0), o posto está vazio. Execute a simulação até que 500 carros, no total, tenham sido atendidos. Estime o tempo de espera médio na fila para os 2 tipos de carro, o número médio de carros na fila para os 2 tipos de carro e a taxa de ocupação da bomba de gasolina.

15) Desenvolva um modelo para um sistema com 2 processos consecutivos (I e II). Os ítens chegam ao sistema com intervalo, médio, entre chegadas de 10 minutos. Assim que chegam, os ítens são imediatamente enviados para o processo I que tem uma fila ilimitada e um recurso simples com uma duração, média, do serviço de 9 minutos. Após terminar o 1º processo, os ítens são enviados para o processo II que é idêntico ao processo I. Após o serviço do processo II ser completado, os ítens deixam o sistema. As medidas de interesse no sistema são o número médio de ítens na fila, em cada processo, e o tempo total, médio, que um ítem permanece no sistema. Usando 10.000 minutos como a duração a ser estudada, faça as 4 simulações a seguir e compare os resultados.

  a) Intervalo entre chegadas exponencial e duração do serviço exponencial.
  b) Intervalo entre chegadas exponencial e duração do serviço constante.
  c) Intervalo entre chegadas constante e duração do serviço exponencial.
  d) Intervalo entre chegadas constante e duração do serviço constante.

16) Peças chegam a uma estação de trabalho com um intervalo, médio, entre chegadas de 21 segundos (exponencial). Após a chegada, as peças são processadas. O tempo de processamento segue uma distribuição triangular com parâmetros 16, 19 e 22. Existem características visuais que determinam se uma peça tem um eventual problema de qualidade. Estão peças, que são cerca de 10% do total, são enviadas para outra estação onde sofrem uma rigorosa inspeção. As demais (90%) são enviadas para a expedição e consideradas boas.
A distribuição do tempo de inspeção rigorosa é, em média, igual a 95 segundos mais uma variável aleatória que segue uma distribuição de Weibull com parâmetros iguais a 48,5 e 4,04. Em torno de 14% das peças que sofrem inspeção, são reprovadas e viram sucata. As demais vão para a expedição.
Execute a simulação para 10.000 segundos para determinar o número de peças boas, o número de peças sucateadas e o número de peças que são inspecionadas parcialmente e rigorosamente.

17) Clientes chegam a uma caixa com um intervalo, médio, entre chegadas igual a 10 minutos (exponencial). Um único funcionário recebe o pagamento além de conferir o pedido. Ele demora, em média, entre 8 a 10 minutos para fazer estas tarefas, variando a duração uniformemente naquele intervalo. Após esta atividade estar completada, o cliente é, aleatoriamente, atribuído a um de 2 funcionários do estoque que separam e embalam a mercadoria para entregar ao cliente. O tempo desta atividade é também uniformemente distribuído entre

16 e 20 minutos. Cada um dos 2 funcionários do estoque só podem atender clientes que foram designados para ele. Após receber sua mercadoria, o cliente vai embora.
Desenvolva um modelo e rode a simulação para 5.000 minutos.
Há uma sugestão de que os 2 funcionários possam atender qualquer cliente que ficariam em uma fila única esperando atendimento. Rode a simulação também para 5.000 minutos e compare os resultados.

Apêndice A

Tabela do χ^2

ν $\Downarrow$	**0,25**	**0,10**	**0,05**	**0,025**	**0,01**	**0,005**	**0,001**	$\Leftarrow \alpha$
1	1,32	2,70	3,84	5,02	6,63	7,87	10,82	
2	2,77	4,60	5,99	7,37	9,21	10,59	13,81	
3	4,10	6,25	7,81	9,34	11,34	12,83	16,26	
4	5,38	7,77	9,48	11,14	13,27	14,86	18,46	
5	6,62	9,23	11,07	12,83	15,08	16,75	20,51	
6	7,84	10,64	12,59	14,44	16,81	18,54	22,45	
7	9,03	12,01	14,06	16,01	18,47	20,27	24,32	
8	10,21	13,36	15,50	17,53	20,09	21,95	26,12	
9	11,38	14,68	16,91	19,02	21,66	23,58	27,87	
10	12,54	15,98	18,30	20,48	23,20	25,18	29,58	
11	13,70	17,27	19,67	21,92	24,72	26,75	31,26	
12	14,84	18,54	21,02	23,33	26,21	28,30	32,90	
13	15,98	19,81	22,36	24,73	27,68	29,81	34,52	
14	17,11	21,06	23,68	26,11	29,14	31,31	36,12	
15	18,24	22,30	24,99	27,48	30,57	32,80	37,69	
16	19,36	23,54	26,29	28,84	32,00	34,26	39,25	
17	20,48	24,76	27,58	30,19	33,40	35,71	40,79	
18	21,60	25,98	28,86	31,52	34,80	37,15	43,31	
19	22,71	27,20	30,14	32,85	36,19	38,58	43,82	
20	23,82	28,41	31,41	34,17	37,56	39,99	45,31	
21	24,93	29,61	32,67	35,47	38,93	41,40	46,79	
22	26,03	30,81	33,92	36,78	40,28	42,79	48,26	
23	27,14	32,00	35,17	38,07	41,63	44,18	49,72	
24	28,24	33,19	36,41	39,36	42,98	45,55	51,17	
25	29,33	34,38	37,65	40,64	44,31	46,92	52,62	
26	30,43	35,56	38,88	41,92	45,64	48,29	54,05	
27	31,52	36,74	40,11	43,19	46,96	49,64	55,47	
28	32,62	37,91	41,33	44,46	48,27	50,99	56,89	
29	33,71	39,08	42,55	45,72	49,58	52,33	58,30	
30	34,80	40,25	43,77	46,97	50,89	53,67	59,70	
40	45,61	51,80	55,75	59,34	63,69	66,76	73,40	
50	56,33	63,16	67,50	71,42	76,15	79,49	86,66	

Apêndice B

Tabela do t de *Student*

ν $\Downarrow$	**0,995**	**0,99**	**0,975**	**0,95**	**0,90**	$\Leftarrow 1-\alpha/2$
1	63,66	31,82	12,71	6,31	3,08	
2	9,92	6,92	4,30	2,92	1,89	
3	5,84	4,54	3,18	2,35	1,64	
4	4,60	3,75	2,78	2,13	1,53	
5	4,03	3,36	2,57	2,02	1,48	
6	3,71	3,14	2,45	1,94	1,44	
7	3,50	3,00	2,36	1,90	1,42	
8	3,36	2,90	2,31	1,86	1,40	
9	3,25	2,82	2,26	1,83	1,38	
10	3,17	2,76	2,23	1,81	1,37	
11	3,11	2,72	2,20	1,80	1,36	
12	3,06	2,68	2,18	1,78	1,36	
13	3,01	2,65	2,16	1,77	1,35	
14	2,98	2,62	2,14	1,76	1,34	
15	2,95	2,60	2,13	1,75	1,34	
16	2,92	2,58	2,12	1,75	1,34	
17	2,90	2,57	2,11	1,74	1,33	
18	2,88	2,55	2,10	1,73	1,33	
19	2,86	2,54	2,09	1,73	1,33	
20	2,84	2,53	2,09	1,72	1,32	
21	2,83	2,52	2,08	1,72	1,32	
22	2,82	2,51	2,07	1,72	1,32	
23	2,81	2,50	2,07	1,71	1,32	
24	2,80	2,49	2,06	1,71	1,32	
25	2,79	2,48	2,06	1,71	1,32	
26	2,78	2,48	2,06	1,71	1,32	
27	2,77	2,47	2,05	1,70	1,31	
28	2,76	2,47	2,05	1,70	1,31	
29	2,76	2,46	2,04	1,70	1,31	
30	2,75	2,46	2,04	1,70	1,31	
40	2,70	2,42	2,02	1,68	1,30	
60	2,66	2,39	2,00	1,67	1,30	
∞	2,58	2,33	1,96	1,64	1,28	

Bibliografia

[1] Averril M. Law, *Simulation Modeling and Analysis*, 4ª Edition, McGraw-Hill, 2007.

[2] W. David Kelton, Randall P. Sadowski and David T. Surrock, *Simulation with Arena*, 4ª Edition, McGraw-Hill, 2007.

[3] Paulo José de Freitas Filho, *Introdução a Modelagem e Simulação de Sistemas*, Visual Books, 2001.

[4] Kevin Watkins, *Discret Event Simulation in C*, McGraw-Hill, 1993.

[5] Jerry Banks, John S. Carson II and Barry L. Nelson, *Discret-Event System Simulation*, 4ª Edition, Prentice Hall, 2004.

[6] James R. Evans and David L. Olsen, *Introduction to Simulation and Risk Analysis*, Prentice-Hall, 1998.

[7] Donald E. Knuth, *The Art of Computer Programming*, Volume 2, Addison-Wesley, 1998.

[8] Clovis Perin Filho, *Introdução à Simulação de Sistemas*, Editora da Unicamp, 1995.

[9] Thomas J. Schriber, *An Introduction to Simulation*, John Wiley & Sons, 1991.

[10] Byron J.T. Morgan, *Elements of Simulation*, Chapman and Hall, 1984.

[11] Michael Pidd, *Computer Simulation in Management Science*, John Wiley & Sons, 1984.

[12] Leonardo Chwif e Afonso C. Medina, *Modelagem e Simulação de Eventos Discretos*, Bravarte, 2006.

[13] Hugh J. Watson and John H. Blackstone Jr., *Computer Simulation*, 2ª Edition, John Wiley & Sons, 1989.

[14] Zaven A. Karian and Edward J. Dudewicz, *Modern Statistical, Systems and GPSS Simulation*, 2ª Edition, CRC Press, 1999.

[15] James A. Chisman, *Introduction to Simulation Modeling using GPSS*, Prentice-Hall, 1992.