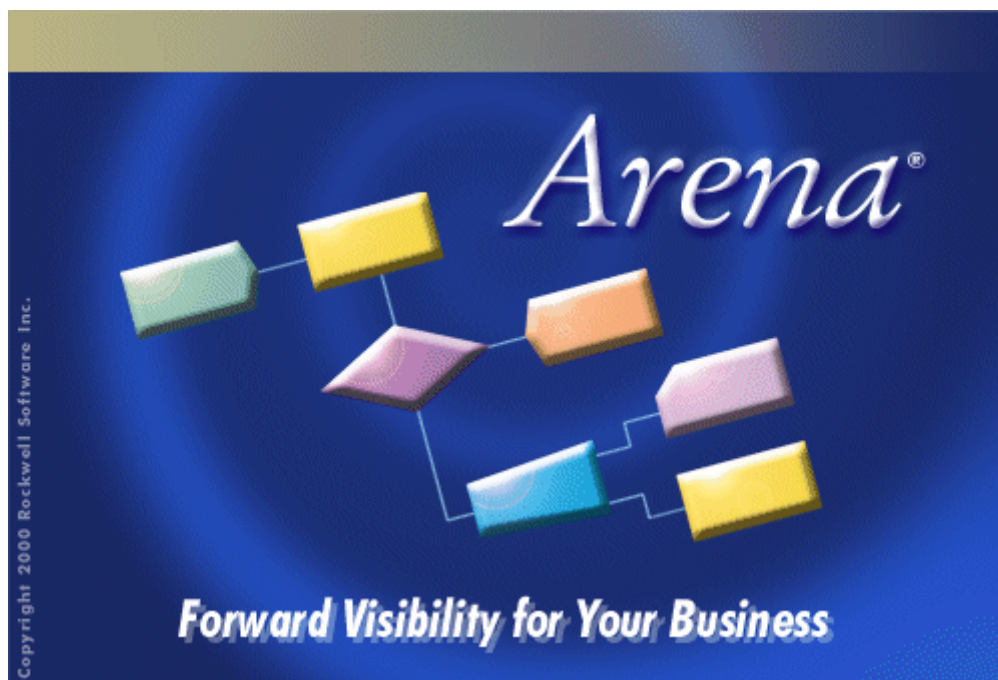


UFSC - UNIVERSIDADE FEDERAL DE SANTA CATARINA  
DAS - DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS  
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO  
DAS5313 - AVALIAÇÃO DE DESEMPENHO DE SISTEMAS  
Professor Ricardo J. Rabelo

## Manual do Arena 9.0



### **Autores:**

**Rodrigo Zago de Lima (Fevereiro/2003, Versão 1.0)**

**Alisson D. C. de Souza (Março/2004, Versão 1.1)**

**Luciane Calixto de Araújo (Agosto/2006, Versão 2.0)**

## Sumário

1.	ARENA? .....	5
2.	Conceitos básicos de modelagem com ARENA 9.0 .....	6
2.1.	Blocos de Modelagem .....	6
2.2.	Entidade e Atributos .....	7
2.3.	Filas.....	7
2.4.	Recursos.....	8
2.5.	Estatísticas (Statistics) .....	8
2.6.	Conjuntos/Grupamentos (Sets) .....	9
2.7.	Estações (Stations) .....	9
2.8.	Armazéns (Storages).....	10
2.9.	Transportadores Fixos ou Condutores (Conveyors).....	10
2.10.	Transportadores Livres (Transporters) .....	10
3.	Definindo “Tempos” .....	11
3.1.	Distribuições Estatísticas.....	11
3.2.	O Input Analyser.....	14
4.	Os Templates do Painel de Projeto .....	16
4.1.	Basic Process .....	16
4.1.1.	O Bloco Create .....	16
4.1.2.	O Bloco Process.....	17
4.1.3.	O Bloco Dispose .....	17
4.1.4.	O Bloco Assign .....	17
4.1.5.	O Bloco Batch .....	18
4.1.6.	O Bloco Decide.....	19
4.1.7.	O Bloco Record .....	20
4.1.8.	O Bloco Separate.....	21
4.1.9.	O Bloco de Dados Entity .....	21
4.1.10.	O Bloco Queue .....	22
4.1.11.	O Bloco Resource .....	22
4.1.12.	O Bloco Variable.....	22
4.1.13.	O Bloco Set .....	22
4.1.14.	O Bloco Schedule. ....	22
4.2.	Advanced Process .....	23
4.2.1.	O Bloco Delay .....	23
4.2.2.	O Bloco Dropoff.....	24
4.2.3.	O Bloco Hold.....	25
4.2.4.	O Bloco Match.....	26
4.2.5.	O Bloco Pickup .....	27
4.2.6.	O Bloco ReadWrite.....	28
4.2.7.	O Bloco Release .....	29
4.2.8.	O Bloco Remove.....	30
4.2.9.	O Bloco Seize.....	31
4.2.10.	O Bloco Search .....	32
4.2.11.	O Bloco Signal .....	33
4.2.12.	O Bloco Store .....	34
4.2.13.	O Bloco Unstore .....	34
4.2.14.	Adjust Variable .....	35
4.2.15.	Advanced Set .....	35
4.2.16.	Expression .....	35
4.2.17.	Failure .....	35
4.2.18.	File.....	35
4.2.19.	State Set .....	35
4.2.20.	Statistic .....	36

4.2.21.	Storage.....	36
4.3.	Advanced Transfer .....	36
4.3.1.	O Bloco Enter.....	36
4.3.2.	O Bloco Leave.....	37
4.3.3.	O Bloco PickStation.....	38
4.3.4.	O Bloco Route.....	39
4.3.5.	O Bloco Station .....	39
4.3.6.	O Bloco Access.....	40
4.3.7.	O Bloco Convey: .....	41
4.3.8.	O Bloco Exit.....	41
4.3.9.	O Bloco Start .....	42
4.3.10.	O Bloco Stop.....	42
4.3.11.	O Bloco Activate.....	43
4.3.12.	O Bloco Allocate .....	44
4.3.13.	O Bloco Free .....	45
4.3.14.	O Bloco Halt .....	45
4.3.15.	O Bloco Move.....	46
4.3.16.	O Bloco Request.....	46
4.3.17.	O Bloco Transport.....	47
4.3.18.	O bloco Sequence.....	48
4.3.19.	O bloco Conveyor .....	48
4.3.20.	O bloco Segment.....	48
4.3.21.	O bloco Transporter .....	49
4.3.22.	O bloco Distance .....	49
4.3.23.	O bloco Network.....	49
4.3.24.	O bloco Network Link .....	50
4.3.25.	O bloco Activity Área.....	50
4.3.26.	Diferenças entre blocos .....	50
5.	Outros recursos de Modelagem .....	52
5.1.	Submodelos .....	52
5.2.	Diferentes “Vistas” para o Modelo.....	52
5.3.	O Painel de Navegação .....	52
6.	Rodando a Simulação e Vendo os Resultados .....	53
6.1.	Preparando uma Seção de Simulação.....	53
6.2.	Iniciando e Terminando Seções de Simulação .....	53
6.3.	Setup.....	54
6.3.1.	Parâmetros de Projeto.....	54
6.3.2.	Parâmetros de Replicação.....	54
6.3.3.	Velocidade de Simulação .....	56
6.3.4.	Relatórios.....	56
7.	Identificando e Corrigindo Erros .....	57
7.1.	Janela Errors/Warnings.....	57
7.2.	Barra de Debug .....	57
7.2.1.	Breakpoints .....	57
7.2.2.	Calendar Window .....	58
7.2.3.	Activitie Entities Window.....	58
7.2.4.	Watch Windows.....	58
8.	Interpretando Resultados/Relatórios .....	59
8.1.	Tipos de cálculos de estatísticas .....	59
8.2.	Contadores e Saídas.....	59
8.3.	Variáveis referentes a Entidades .....	60
8.4.	Variáveis referentes a Filas.....	60
8.5.	Variáveis referentes a Recursos .....	60
8.6.	Variáveis referentes a Condutores .....	61
8.7.	Variáveis referentes a Transportadores .....	61
8.8.	Variáveis referentes a Processos .....	61
9.	O Process Analyser .....	62

9.1.	Definição de Termos .....	62
9.2.	Procedimento Geral para Análise de Cenários .....	62
9.3.	Exemplo do Funcionamento do Process Analyser .....	63
9.3.1.	Iniciando a Ferramenta .....	63
9.3.2.	Criando um novo projeto .....	63
9.3.3.	Adicionando um cenário .....	63
9.3.4.	Adicionando Controles .....	64
9.3.5.	Adicionando Respostas .....	65
9.3.6.	Executando o cenário .....	65
9.3.7.	Apontando suas “Soluções” .....	65
9.3.8.	Transformando seus dados de saída em gráficos .....	66
10.	Animação com ARENA 9.0 .....	67
10.1.	Animação de Fluxo .....	67
10.1.1.	Entidades .....	67
10.1.2.	Biblioteca .....	67
10.2.	Animação com a barra de ferramentas de animação básica .....	68
10.2.1.	Clock: .....	68
10.2.2.	Date .....	69
10.2.3.	Variable .....	69
10.2.4.	Level .....	70
10.2.5.	Histogram .....	70
10.2.6.	Plot .....	71
10.2.7.	Queue .....	71
10.3.	Animação de blocos de transferência avançados .....	72
10.3.1.	Storage .....	72
10.3.2.	Parking Area .....	72
10.3.3.	Seize Area .....	73
10.3.4.	Transporter .....	73
10.3.5.	Station .....	73
10.3.6.	Intersection .....	73
10.3.7.	Route .....	74
10.3.8.	Segment .....	74
10.3.9.	Distance .....	74
10.3.10.	Network .....	75
10.3.11.	Promote Path .....	75
10.4.	Fábrica de símbolos ARENA .....	75
10.4.1.	Resource e transporter .....	75
10.4.2.	Global Picture Placement .....	76

# 1. ARENA?

A resposta à pergunta “O que é o ARENA ?” é simples. O Arena é um software de simulação de processos.

A simulação de processos é uma das ferramentas para estudo de sistemas podendo ser feita através do uso de protótipos ou pela elaboração de modelos matemáticos. A utilização de modelos matemáticos nos permite encontrar soluções analíticas ou implementar modelos para simulação. O ARENA se enquadra no último caso, fornecendo uma interface gráfica que permite a elaboração de um modelo de simulação baseado na linguagem SIMAN.

Ainda cabe a pergunta, mas porque simular processos? A simulação de processos é uma ferramenta eficaz no auxílio à tomada de decisões dentro da empresa. Ele permite criar e testar experimentos em modelos dos diversos sistemas de uma corporação. Testando novas idéias e projetos em um simulador, você pode “predizer” o que acontecerá, sem ter que efetivamente alterar qualquer coisa no ambiente físico. No caso do ARENA as vantagens oferecidas, beneficiam desde as tomadas de decisão sobre chão-de-fábrica, até o front-office da empresa, podendo a simulação ser usada em qualquer tipo de ambiente seja ele um pátio de mineração ou uma U.T.I de hospital.

Os principais passos de uma modelagem utilizando o software seriam:

## **Criar um modelo básico**

Isso se faz “arrastando” os blocos lógicos que representam processos decisórios, criação de entidades, métodos de transporte e outros para dentro da área de projeto. Os blocos são conectados de forma a seguirem um fluxo de informação que representa abstratamente o processo real a ser modelado. O ARENA ainda permite o estabelecimento de uma estrutura de informação complexa que permite armazenar variáveis e, através dessas, definir comportamentos para o sistema.

## **Refinar o modelo**

Através das ferramentas de acompanhamento de simulação é possível verificar seu modelo identificando erros lógicos e melhorar a modelagem de forma a obter um modelo claro e conciso. Uma vez finalizado o modelo lógico do sistema, caso seja de interesse, é possível elaborar uma animação que permita visualizar mais claramente todos os elementos do sistema.

## **Simular o modelo**

Terminada a etapa de modelagem, passa-se a simulação. Nessa etapa será possível verificar se realmente reflete o sistema. Uma vez que o modelo seja válido, é hora de dar início às simulações, utilizando diferentes cenários de forma identificar melhorias.

## **Analisar os resultados da simulação**

Feita a simulação, o ARENA gera relatórios automáticos com dados específicos como taxa de utilização dos recursos ou tempos de espera. Durante a fase de modelagem é possível criar novos dados estatísticos de acordo com o que seja interessante para a análise, customizando os relatórios do Arena de forma que eles se adaptem ao que é realmente necessário.

## **Selecionar a melhor alternativa**

Baseado nos resultados da análise é possível selecionar a melhor solução para o seu problema e testa-la exaustivamente.

## 2. Conceitos básicos de modelagem com ARENA 9.0

O ARENA tem uma “visão de mundo” bastante peculiar, baseada na simplificação através de eventos discretos. Os modelos são baseados na linguagem de simulação SIMAN. Essa linguagem basicamente enxerga o sistema como uma seqüência de eventos aleatórios que causam mudanças nos estado do modelo. De uma forma um pouco grosseira, a modelagem com ARENA assemelha-se à modelagem através de Redes de Petri ou Autômatos.

Assim, os sistemas normalmente são reduzidos a um conjunto de estações de trabalho que prestam serviços a clientes (entidades). Cada bloco ou módulo é interconectado a outros e as entidades se locomovem entre eles seguindo uma seqüência lógica. Podemos assim fazer uma analogia com várias situações a serem modeladas, como por exemplo:

Pessoas (entidades) percorrendo as diversas seções (estações) de um supermercado onde efetuam compras;

Um automóvel (entidade) sendo fabricado nas diversas seções (estações) de uma fábrica;

Uma apólice de seguro (entidade) sendo processada nas diversas seções (estações) de uma seguradora;

O caminho da entidade pelo sistema pode depender de vários atributos da própria entidade ou variáveis dentro do sistema.

Por exemplo, um carro sendo montado em uma fábrica pode seguir para uma ou outra estação de trabalho dependendo qual o modelo do carro, ou do “Schedule” de trabalho da equipe fabril. Na seqüência são apresentados alguns conceitos que vão facilitar a compreensão da modelagem com ARENA.

### 2.1. Blocos de Modelagem

Os modelos ARENA são construídos através de blocos de modelagem. A princípio, falaremos de dois tipos de módulos ou blocos: os módulos de fluxo e os módulos de dados. Ambos estão representados nas figuras abaixo.



Figura 1 - Módulos de Fluxo Básicos

Os módulos de Fluxo são aqueles que realmente são interconectados e formam uma rede de informações e comandos por onde as entidades seguirão. Eles descrevem a lógica do processo da maneira mais visual possível.



**Figura 2 - Módulos de Dados Básicos**

Os módulos de dados não aparecem diretamente no modelo, ao contrário, eles são editados através de formulários. Sua função é justamente inserir as especificações de cada elemento do fluxo, mas não a de ter caráter ativo na simulação.

Os módulos são acessados através do Painel de Navegação a direita na interface.

## **2.2. Entidade e Atributos**

Entidades representam qualquer objeto que se mova através do sistema, e por objeto entenda-se tanto os “reais” como abstratos, por exemplo, informação. Cada entidade tem características próprias que as definem e são denominadas atributos.

Os atributos servem como “etiquetas” ou “código de barras” que estão estampados na entidade e podem ser usados na hora de tomadas de decisões dentro do fluxo. Quando um atributo é definido, mesmo que o seu valor altere-se durante o “caminho” da entidade no fluxo, a mudança é referente apenas àquela entidade específica. Para alterações em valores de utilização global devem-se utilizar “variáveis” e não atributos.

Como exemplo, temos atributos que já “nascem” com a entidade como seu número de série (Entity.SerialNumber) e outros que podem ser definidos quando a entidade passa, como exemplo, por um bloco Assign (ver “Blocos Básico”) tal qual “Tamanho”, “Prioridade”, “Cor”, entre outros.

## **2.3. Filas**

As filas servem, como seria de se esperar, como um reservatório para entidades que não podem continuar seguindo normalmente no fluxo por algum motivo (um recurso ocupado e a espera de uma autorização são exemplos típicos). Entidades entram na fila e são removidas com base na mudança de status do sistema do elemento associado com a mesma.

Existem dois tipos diferentes de filas usadas pelo software: as individuais e as internas. No primeiro caso, a fila possui um nome, um modo de seleção e uma maneira específica de “ranquear” as entidades (First-In, First-out, Highest attribute First, etc), e uma capacidade própria. Entidades em filas individuais podem ser mostradas na animação, suas estatísticas são coletadas e podem ser “ranqueadas”.

Já no caso de filas do tipo internas, elas funcionam basicamente com ranqueamento do tipo “First-in, First-out” e não são animadas, nem coletam estatísticas ou trabalham com outros mecanismos de ranqueamento.

O número de entidades esperando em uma fila pode ser acessado através da seguinte expressão NQ (nome da fila.queue). Essa informação pode ser usada em casos como: um cliente entra em um processo decisório no qual ele deve decidir qual das filas ele tomará, escolhendo por fim aquela com o menor valor gravado no atributo NQ.

## **2.4. Recursos**

Recursos são os elementos que não se “movem”, por assim dizer, no sistema e são alocados para as entidades. Um recurso não é exatamente um objeto imóvel, pode ser um operador que se movimenta de um lado para o outro, mas o importante é que ele não se mova pelo “fluxo lógico” do seu modelo.

Eles possuem capacidade finita (que pode variar com o tempo) e um conjunto de estados (ocupado, em espera, inativo, em falha, etc) que se alteram com o decorrer da simulação à mercê da lógica do sistema modelado.

Para entender as opções que o bloco Resources te oferecerá, deve-se ter em mente a seguinte terminologia: quando a entidade requer um recurso, ela o “reserva” (Seize); e quando, ela já foi processada e não mais necessita do recurso, ela o “libera” (Release), de forma que este possa ser “reservado” por outras entidades. Cada recurso possui uma fila associada para que sejam alocadas as entidades que tentam “reservá-lo”, mas não conseguem porque ele não está disponível no momento. (Veja o item “Filas”)

A capacidade de um recurso refere-se ao número de entidades que podem reservá-lo ao mesmo tempo, enquanto que a quantidade de um recurso refere-se ao número de recursos que serão empregados. Uma outra maneira de enxergar a questão é que a capacidade define processamento em paralelo e quantidade representa processamento em série. Exemplificando: Um recurso “Máquina” com capacidade “2” pode produzir duas peças ao mesmo tempo, enquanto que um recurso “Máquina” com quantidade “2”, quer dizer que o produto será processado em uma máquina e depois na outra subsequentemente.

A capacidade de um recurso pode alterar-se em função do tempo, sendo variável no período de simulação, através da definição de uma programação (schedule) que adicione ou remova unidades do recurso para representar turnos, staff temporário, padrões, etc. Tempos de descanso (Downtimes) ou falhas (Failures) podem também ser associados com os recursos para simular atividades planejadas ou não, como pausas ou manutenção de máquinas.

É importante frisar que um recurso com capacidade superior a 1 (um) é ainda assim representado no modelo por um único recurso para fins de coleta de estatísticas (os dados estatísticos representarão todo o conjunto e não cada parte em separado). Se for necessário possuir a estatística de cada parte, elas devem ser definidas como recursos independentes e reunidas em um Conjunto/Grupamento (SET).

Os recursos são simbolizados na animação geral do modelo como figuras estáticas que se sobrepõem representando cada estado do mesmo (em espera, ocupado, etc.).

## **2.5. Estatísticas (Statistics)**

À medida que o modelo é simulado o ARENA coleta dados referentes a simulação e os armazena em um banco de dados para ao final da simulação gerar um relatório referente a simulação.

Existem quatro tipos de categorias de estatísticas referentes a tempo e custos que são registradas para cada entidade no sistema bem como para cada processos pelo qual a entidade passe. Essas cinco categorias são: wait time/cost, Value-added time/cost, non-value added time/cost, transfer time/cost e other time/cost. Baseados nestas é são calculados custom total e tempo total. As estatísticas referentes a processos e entidades são calculadas separadamente.

Além de estatísticas sobre processos e recursos, há ainda informações sobre Filas e Recursos que incluem como o tempo de utilização de cada recurso, a capacidade ocupada de cada um deles, tempo de espera em filas, etc.



O usuário poderá ainda definir outros dados estatísticos a ser apresentados no relatório de simulação através do módulo Record e de estatísticas persistentes do bloco de dados Variáveis.

Por fim para cada bloco do modelo são calculadas estatísticas referentes ao número de entidades que passam por estes.

É possível especificar se as estatísticas da simulação de um determinado elemento (fila, recurso, etc.) devem ou não ser coletadas. Para tanto basta desmarcar a opção Report Statistics na janela de edição referente ao bloco.

## **2.6. Conjuntos/Grupamentos (Sets)**

Quando se tem um conjunto de elementos na simulação que, apesar de possuírem características individuais específicas, são “acessados” de maneira similar, tais elementos podem ser agrupados em um bloco apenas (um “Set”). O exemplo mais comum é o de vários recursos similares que desempenham a mesma função, mas com características específicas (por exemplo, o tempo de processamento) que são agrupados em um só conjunto e acessados pela entidade como se fosse apenas um elemento.

Imagine que um cliente entre em uma loja e existam várias balconistas para atendê-lo. Ele pode escolher qualquer uma delas, mas cada uma terá uma maneira diferente de fazer o atendimento. As balconistas podem então ser agrupadas num “Set” de nome “Atendentes” e quando a entidade “cliente” entra no módulo de processamento, ela reserva para si uma unidade do recurso “Atendentes” que pode ser qualquer uma das balconistas disponíveis.

Nada impede que um recurso possa participar de mais de um “Set” ao mesmo tempo. Por exemplo, no caso acima, uma das balconistas poderia também ser parte da “Set” denominada “Caixas”, e ficar revezando-se entre o processamento no módulo “Atendimento” e no módulo “Pagamento”.

É importante relatar que podem ser definidos diferentes tipos de conjuntos, incluindo: resource, conter, tally, entity type e entity picture. Sets do tipo recurso podem ser usados nos bloco Process e ainda em alguns blocos avançados e de transferência. Conjuntos do tipo counter e tally podem ser usados no módulo Record. Conjuntos de fila podem ser usados com alguns dos blocos de processo avançado e alguns blocos de transferência (Seize, Hold, Access, Request, Leave e Allocate).

## **2.7. Estações (Stations)**

As estações são utilizadas especialmente para isolar partes do modelo que funcionem independentemente de outras quer seja lógica ou fisicamente. Explicando melhor, pode-se dizer que são utilizadas para fins de “animação avançada” do seu sistema. A idéia é que todos os blocos lógicos que não envolvam movimentação física da entidade (correspondente com o sistema real) sejam agrupados em uma estação. E para que a entidade se desloque de uma estação para outra é necessário que utilize um dos tipos de transporte que o Arena dispõe.

Exemplificando:

Uma peça chega a uma máquina de CNC para ser processada, lá ela sofrerá várias ações (passará por vários módulos na simulação) mas fisicamente não estará movimentando-se, e só quando houver terminado todo o processamento um AGV a levará para o estoque. Nesse caso, todos os blocos utilizados para representar as ações executadas pela máquina podem ser agrupados em uma estação chamada “CNC” e o transporte entre esta estação e a outra, denominada estoque, será feito por um “transportador” que denominaremos de “AGV”.

Assim, sempre que uma entidade chega a uma estação ela é registrada nesta estação e assim permanece até que passe por outra estação.

## **2.8. Armazéns (Storages)**

Os armazéns possuem basicamente a mesma função “abstrata” das estações. De maneira similar, eles isolam uma certa quantidade de blocos lógicos do modelo, mas ao contrário das estações eles não concluem com a movimentação da entidade de um ponto a outro do modelo. Em resumo eles servem para representar, em termos de animação, que a entidade esteve “parada” em algum ponto do sistema por algum tempo, sofrendo ou não vários tipos de ações (tomada de decisões, processamento, etc.), e posteriormente continuando sua movimentação dentro do sistema, ainda que não haja movimentação equivalente da peça no sistema real.

Esse tópico será melhor compreendido quando forem analisados os blocos de processamento avançado Store e Unstore.

## **2.9. Transportadores Fixos ou Condutores (Conveyors)**

Os transportadores fixos são um dos tipos de dispositivos utilizados para se movimentar uma entidade de uma estação para outra. São assim denominados por não se movimentarem em conjunto com a entidade, apenas fazendo com que ela avance. Um exemplo típico desse caso são as esteiras transportadoras.

Esses dispositivos são definidos pelos pares de estações que eles conectam e pela distância entre elas, denominada segmento. As entidades devem ser carregadas sobre ou dentro do transportador fixo em qualquer uma dessas estações para poder então ser levada até a estação destino.

Cada condutor é dividido em várias células que representam, cada uma, a menor unidade de espaço requerido para conter uma entidade apenas. Quando uma entidade é carregada sobre ou dentro do transportador, ela ocupa uma ou mais células, dependendo de seu tamanho, e impede que outras entidades entrem no transportador até que ela tenha se movimentado e novos espaços tenham aparecido. Quando uma entidade pára de movimentar-se, o efeito sobre as outras entidades em movimento depende do tipo de transportador em questão. Condutores acumulativos (Accumulating Conveyors) bloqueiam o progresso das entidades que se aproximam daquela parada, enquanto as outras continuam normalmente a se movimentar. Já no caso de condutores não-acumulativos (Non-Accumulating Conveyors), a movimentação de todas as entidades sobre eles é interrompida quando qualquer uma delas pára.

## **2.10. Transportadores Livres (Transporters)**

Os Transportadores Livres são um outro tipo de recurso utilizado para levar entidades de uma estação a outra no sistema. Eles normalmente representam veículos como AGVs, caminhões e carros, mas podem eventualmente representar pessoas, carregadores, ou qualquer outra coisa do tipo. Informações necessárias para defini-los são a velocidade do dispositivo e a distância entre as estações que ele conecta.

Assim como os recursos, os transportadores podem estar em diferentes estados: em espera, quando aguarda a solicitação de uma entidade para ser movimentada; ocupado, quando está carregando uma entidade de uma estação a outra; inativo, quando houver alguma falha ou ele estiver em manutenção.

Ao contrário dos transportadores fixos, estes se movimentam em conjunto com a entidade e inclusive, na animação, a figura movimentando-se é justamente a do transportador e não a da entidade como ocorre no caso das esteiras.

### 3. Definindo “Tempos”

O tempo entre chegada de peças ou o tempo de atraso causado por um processamento, entre muitos outros, pode ser definido como uma constante (o que normalmente não é muito válido para um processo real), uma distribuição estatística ou uma expressão matemática.

O tipo de dado que você vai usar depende do processo que está modelando, e dos dados disponíveis para ele. Em um simples modelo de teste, ou para um processo constante (o que não é muito comum para processos “reais”), um tempo constante é passível de utilização.

Provavelmente o mais utilizado, em especial para o projeto da disciplina de avaliação de desempenho, são as distribuições estatísticas (detalhadas no próximo item). A partir de dados tomados experimentalmente, podemos gerar curvas aproximadas de probabilidades utilizando o Input Analyser (confira seu funcionamento mais abaixo).

Como exemplo de quando você deveria usar uma expressão matemática, temos a situação em que, por exemplo, o tempo de processamento esteja relacionado de alguma maneira com atributos dados a entidades. Explicitando: peças de diferentes pesos levam tempos diferentes para serem levantadas por um sistema de carregamento seguindo a seguinte expressão:  $(0.5 * \text{Peso\_entidade}) - 1$ .

Ainda é possível definir padrões de tempo para capacidade de recursos e chegada de entidades através da definição de um Time Pattern.

#### 3.1. Distribuições Estatísticas

As distribuições Triangular e Uniforme têm particular importância pelo fato de podermos usá-las quando ainda não efetuamos o levantamento dos dados. No caso da triangular trabalha-se com um número médio, um mínimo e um máximo e a Uniforme com um valor constante entre dois extremos. São utilizadas na montagem de modelos do Arena como uma aproximação intuitiva de distribuições do processo e são, posteriormente, substituídas pelas distribuições definitivas. Cuidados devem ser tomados para evitar a “tentação” de se trabalhar com estas distribuições, em vista da simplicidade de uso. Isto somente deve ser feito quando houver a comprovação com a realidade.

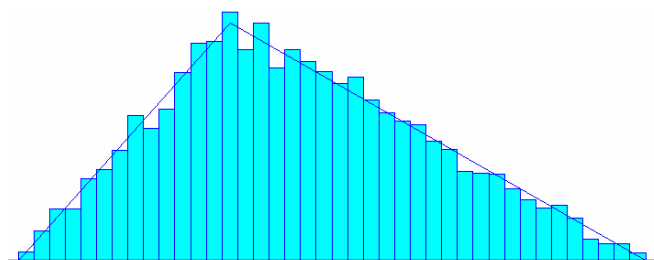


Figura 3 – Histograma de uma distribuição de probabilidades do tipo Triangular

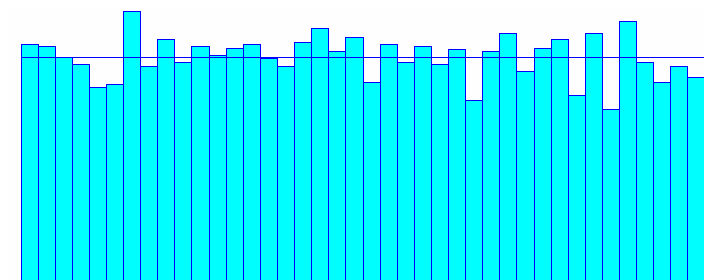
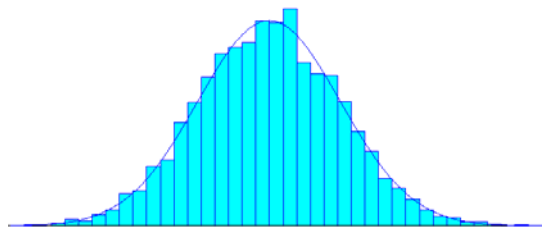


Figura 4 - Histograma de uma distribuição de probabilidades do tipo uniforme

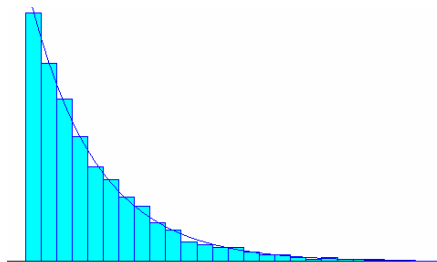
A Normal é um tipo de distribuição “universal”, é a mais adaptável entre todas e a que mais se verifica de maneira geral em casos reais seja para tempos de atendimento, seja para tempos de chegada (com mais ênfase nos primeiros). O motivo é simples: a distribuição normal trabalha com uma média e um desvio padrão, ou seja, com um certo tempo e uma probabilidade de que seja um pouco menor ou um pouco maior.

É fácil imaginar que em grande parte das situações, os processos funcionam dessa maneira. “A peça demora em média 3 minutos para ser manufaturada, um pouco mais ou um pouco menos” ou “Um caixa atende uma pessoa levando cinco minutos em média, às vezes alguns tem mais problemas e demoram um pouco mais e outras vezes tem apenas uma conta para pagar e o atendimento é mais rápido” são análises comuns de tempos no dia-a-dia.



**Figura 5 - Histograma de uma distribuição do tipo Normal**

A distribuição Exponencial é por empirismo a mais representativa em termos de tempos de chegada. Ela é baseada na definição de um tempo mínimo médio, ou seja, as entidades chegam ao sistema em sua grande maioria em um certo intervalo de tempo, algumas poucas chegam antes disso e outras chegam em intervalos um pouco maiores.

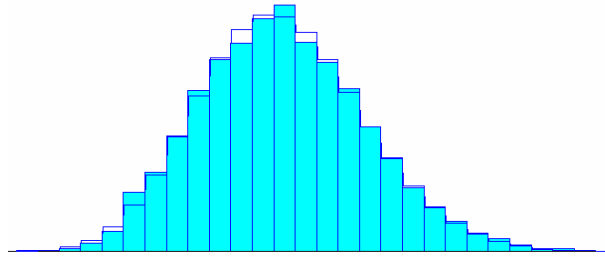


**Figura 6 - Histograma de uma distribuição de probabilidades do tipo Exponencial**

A distribuição Poisson é bastante comum para representar tempos de chegada, contudo ela difere um pouco da Exponencial, pois trabalhar muito mais com “ritmos” de chegada do que com frequências. Por exemplo, se estivermos trabalhando com a chegada de carros a um sistema de pedágio, ou a um posto de gasolina e dissermos que o número de carros que chegam por um determinado intervalo de tempo segue a tabela na próxima página:

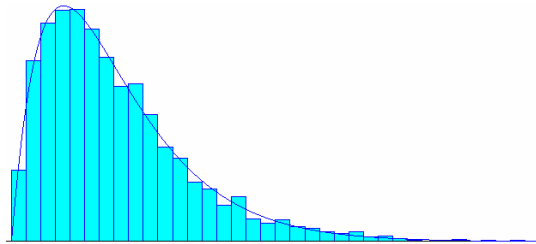
2	0	2	1	0	2	1	0	1	2
1	2	3	1	3	1	3	4	5	1
2	0	1	2	1	0	1	1	0	2
2	2	3	2	2	3	2	3	3	2

Se colocarmos esses dados no Input Analyser e pedirmos para que se verifique qual a melhor distribuição probabilística que se encaixa a ele, teremos como resultado uma Poisson como a da figura abaixo.

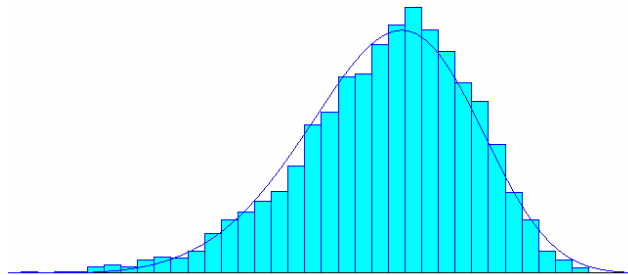


**Figura 7 - Histograma de uma distribuição de probabilidades do tipo Poisson**

No que se refere a tempos de atendimento, distribuições muito parecidas com a Exponencial, mas com mais parâmetros e mais bem elaboradas matematicamente como a Erlang, a Gamma ou a Weibull se mostram bastante eficientes. Elas inserem fatores como possíveis “alargamentos” da média de valores, ou uma maior representatividade dos valores inferiores à média ao sistema.



**Figura 8 - Histograma de uma distribuição do tipo Erlang**



**Figura 9 - Histograma de uma distribuição do tipo Erlang**

Por fim, as distribuições Discretas e Contínuas são utilizadas para se colocar dentro do sistema a “distribuição verdadeira”, ou seja, os dados exatos que foram obtidos através de pesquisa em campo. Esse procedimento só é válido se os dados forem em quantidade muito grande e extremamente significativos com relação ao processo. Mas de qualquer forma, sua utilização é tão trabalhosa que se torna ineficiente. De maneira geral, pode ser utilizado para testar a eficiência das outras distribuições quando trabalhando com pequenos intervalos de tempo.

No quadro abaixo, segue um pequeno resumo sobre as distribuições de frequência:

<b>Distribuição</b>	<b>Abreviatura</b>	<b>Parâmetros</b>	<b>Melhor Aplicação</b>
Poisson	POIS	Média	Chegada
Exponencial	EXPO	Média	Chegada
Triangular	TRIA	Min/ Média/Max	Atendimento
Uniforme	UNIF	Min/ Média/Max	Atendimento
Normal	NORM	Média/Desvio	Atendimento
Johnson	JOHN	G, D, L, X	Atendimento
Log Neperiano	LOGN	Média Logrtm.	Atendimento
Weibull	WEIB	Beta, Alfa	Atendimento
Discreta	DISC	P1, V1, ...	Cheg/Atendim
Contínua	CONT	P1, V1, ...	Cheg/Atendim
Erlang	ERLA	Média / K	Atendimento
Gamma	GAMM	Beta, Alfa	Atendimento

### 3.2. O Input Analyser

O Input Analyser (IA) é uma ferramenta padrão oferecida no ambiente Arena. Ela é utilizada para se determinar a qualidade do “casamento” entre as funções de distribuição de probabilidades com os dados de entrada coletados em campo. A ferramenta pode ser acessada através do Menu **Tools->Input Analyser**.

Os dados processados pelo IA são normalmente os intervalos de tempo entre começo e fim de processamento, ou entre chegadas de entidades ao sistema e outros tempos aleatórios.

Uma maneira de “conhecer” as diferentes opções de distribuições probabilísticas oferecidas pelo IA, você pode simplesmente gerá-las a partir do comando **File->DataFile->Generate New...**

Você escolhe qual das distribuições você quer verificar, quais seus parâmetros e logo em seguida um gráfico é apresentado. Foi este comando que deu origem aos exemplos da seção anterior, “Distribuições Estatísticas”, desse manual.

Para preparar um arquivo com seus dados a ser analisado pelo IA, simplesmente utilize qualquer editor de texto, de preferência um mais simples como o bloco de notas, e separe os diferentes intervalos de tempo por espaços em branco ou descendo a linha após cada inserção de dado. Caso use o Word, salve o arquivo no formato “text only” para eliminar qualquer tipo de formatação indesejada. Então ao invés de escolher a opção “Generate New” como mostrado na figura acima, escolha “Use Existing”.

Você terá então um gráfico de todos os seus pontos coletados já separados em classes e caso você queira que os intervalos de aquisição do seu histograma sejam maiores ou menores, basta selecionar a opção de menu **Options->Parameters-> Histogram...** Conforme mostrado na figura 57.

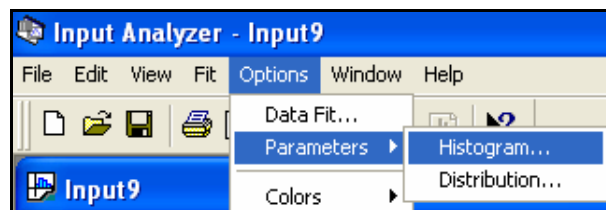


Figura 10 - Acesso aos parâmetros de um histograma

O que você quer saber então é qual curva de probabilidade representa melhor aquela distribuição e quais são os seus parâmetros. Para isso, utiliza a opção **Fit->Fit All**:

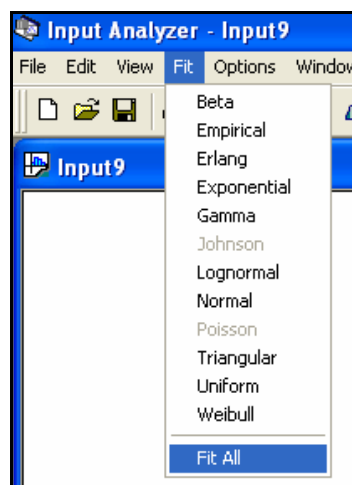


Figura 11 - Acesso a comando Fit All para ajuste de dados

Assim, na parte inferior do painel aparecerá o nome da distribuição e seus parâmetros. Serão esses dados então que você utilizará como dados de entrada do seu modelo. Como mostrado na figura 59.

```
Distribution Summary  
  
Distribution:    Lognormal  
Expression:    14 + LOGN(6.01, 2.29)  
Square Error:  0.001818  
  
Chi Square Test  
Number of intervals = 30  
Degrees of freedom  = 27  
Test Statistic      = 322  
Corresponding p-value < 0.005  
  
Kolmogorov-Smirnov Test  
Test Statistic      = 0.0478  
Corresponding p-value < 0.01
```

Figura 12 - Resultado do ajuste de dados feito com Input Analyzer

## 4. Os Templates do Painel de Projeto

Para a construção de qualquer modelo, o primeiro passo será definir que conjunto de templates será útil para aquela modelagem, isso, obviamente, depois de ter disponibilizado esses templates. Para tanto:

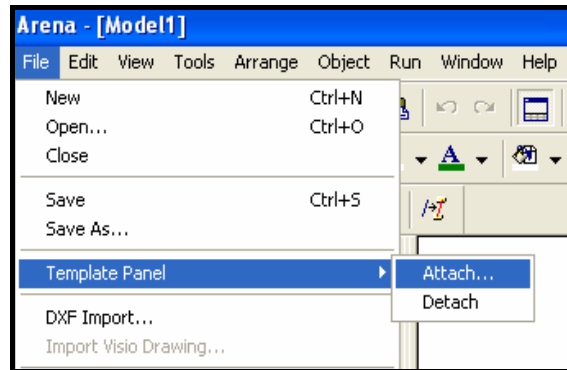


Figura 13 - Disponibilizando os Conjuntos de Blocos

Um *Template* inclui dois tipos de blocos, os lógicos e os de dados. No primeiro caso, você os arrasta com o mouse para a área de projeto e forma o seu fluxo de informação interconectando-os, enquanto que no segundo, ao clicar sobre eles, abaixo da área de projeto irá aparecer uma sequência de colunas com as informações sobre as entidades, recursos, transportadores e outros presentes até o momento em seu projeto.

### 4.1. Basic Process

Os blocos básicos são a estrutura principal na criação de um modelo. Sistemas mais simples, em que podem ser menosprezadas as distâncias entre processos, ou seja, os tempos em que não existam “valor adicionado” podem ser considerados instantâneos, são passíveis, via-de-regra, de serem modelados simplesmente com os blocos básicos. A seguir segue uma breve explicação de cada um dos blocos e algumas aplicações:

#### 4.1.1. O Bloco Create



Create

O ponto de partida para todo modelo é o bloco Create. É desse ponto que as entidades surgem no sistema, sejam elas clientes, documentos, peças, equipamentos, navios, aviões, qualquer coisa. As entidades são tudo aquilo que “sofrerá” a ação das diversas operações lógicas do fluxo.

As entidades são o “gatilho” dos processos, depois de definidas a maneira como são criadas, elas se movem pelo sistema e fazem os processos efetivamente funcionar.

No bloco create são inseridas duas informações importantes relativas ao modelo: os intervalos de tempo em que são criadas as entidades e o tipo das entidades. O tipo da entidade é um atributo da mesma que permite identificar um determinado grupo de entidades e pode servir como parâmetro para decisões lógicas do modelo. Maiores informações sobre os intervalos de tempo podem ser encontradas na seção “Definindo Tempos”.



#### 4.1.2.O Bloco Process



Process

Este é o bloco mais simples para ser usado quando uma entidade passa por alguma ação envolvendo um intervalo de tempo e/ou recursos, por exemplo, um cliente sendo atendido por um funcionário, uma peça sofrendo usinagem, um formulário sendo preenchido por algum funcionário ou outra coisa do tipo.

Vamos usar como exemplo a entidade “peça” seguindo pelo fluxo do sistema. Ela deve em dado instante sofrer um processo de furação. Quando estiver na máquina, obviamente outra peça que estiver vindo “atrás” dela terá que esperar em uma fila antes de ser processada. Para isso devemos selecionar a ação lógica “Seize Delay Release”. De forma que a peça que entra reserva (seize) a máquina para si, sofre o processamento, representado por um certo atraso no seu prosseguimento no fluxo (delay) e depois libera (release) a furadeira para que a próxima peça possa ser processada.

Uma outra possibilidade é que nós tenhamos uma peça que precisa sofrer três processos consecutivos e que todos sejam, por exemplo, em uma mesma máquina, de forma que outra peça não possa ser processada antes que a última tenha passado por toda a seqüência. Para isso usaremos as outras opções de ação lógica.

No primeiro processo, selecionamos então “Seize Delay” para que a máquina seja reservada e utilizada, mas que quando a entidade deixe-a para o próximo passo, ela não esteja livre para que outra peça entre. No segundo processo, um simples “Delay” resolve a modelagem e no terceiro utilizaremos, portanto, “Delay Release” e a seqüência de processos estará então liberada para o uso de uma próxima entidade.

É importante aqui lembrar da diferença entre os conceitos de capacidade e quantidade. Ao determinar que determinado processo precisa de uma quantidade maior do que 1 de um determinado recurso significa que o processo esperará até que a quantidade determinada do recurso esteja disponível. Assim, um processo que precisa de dois funcionários para acontecer, pode ser modelado como precisando da quantidade 2 do recurso funcionários sendo que o modelo deve comportar uma capacidade maior que 2 para o recurso funcionário.

Também existe a opção de modelar o recurso funcionário como um conjunto (set) de funcionários de forma que será possível coletar estatísticas individuais para cada um dos funcionários. Assim, caso existam, por exemplo, três funcionários e sejam necessário dois para executar um determinado processo, o ARENA escolherá dois deles dentro do grupo e alocará para o processo.

#### 4.1.3.O Bloco Dispose



Dispose

Esse bloco é o “fim” de todo projeto de simulação. É por ele que as entidades desaparecem do sistema, sendo tão obrigatório quanto o bloco Create. Se existem muitas entidades acumulando-se na fila de um processo pode ter ocorrido um erro de modelagem em que uma entidade entrou no bloco Dispose sem antes ter liberado o processo que havia reservado.

#### 4.1.4.O Bloco Assign



Assign

Este bloco é utilizado quando se quer trocar o valor de uma variável, “etiquetar” alguma entidade com um atributo específico, trocar a figura representativa da entidade que percorre o fluxo de informações, enfim, alterar o valor de algum parâmetro ou variável do modelo.

Por exemplo, se uma entidade passa por um bloco Assign que está configurado com os seguintes parâmetros:

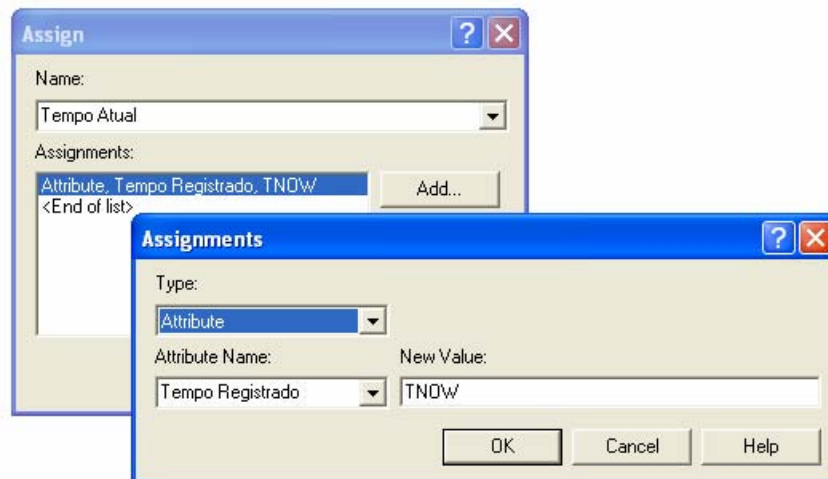


Figura 14 - Exemplo de utilização do Assign: Attribute

O atributo “Tempo Registrado” será modificado recebendo o valor corrente do tempo de simulação. Esse atributo pode posteriormente ser utilizado fazendo-se a entidade passar por um bloco Record e usando estatísticas de intervalo. Dessa maneira pode-se saber o tempo que a entidade demorou desde o momento em que passou pelo bloco assign até o momento em que entrou no bloco Record.

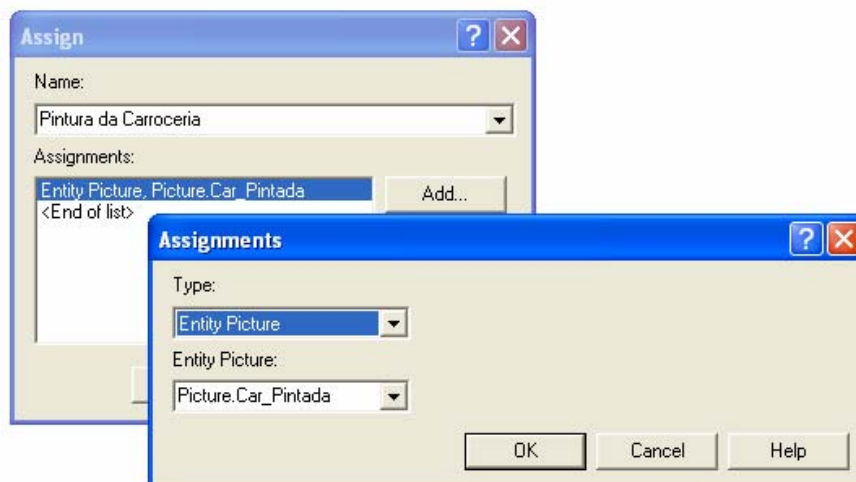


Figura 15 - Exemplo de utilização do Assign: Entity.Picture

Nesse outro exemplo, teríamos a mudança da figura que representa a entidade no sistema. Um caso razoável é alguma matéria-prima chegando a um processo e saindo logo depois como um produto acabado, a carroceria de um carro chegando em uma estação de pintura e depois saindo pintada, etc.

#### 4.1.5.O Bloco Batch



Batch

Esse bloco é utilizado como um mecanismo de agrupamento de entidades que podem posteriormente ser separadas ou não posteriormente através do bloco Separate. O agrupamento é feito através de contagem simples, do tipo “as primeiras quatro entidades que chegam ao bloco, se agrupam e seguem o fluxo e as próximas quatro formam mais um grupo e etc.”, ou então baseado em um atributo.

As entidades que chegam a esse bloco ficam esperando em uma fila até que o número, com ou sem atributo específico, requerido para o agrupamento seja atingido, após passarem pelo bloco, as entidades agrupadas seguem pelo fluxo com se fossem uma só nova entidade.

Imaginemos, por exemplo, que exista um AGV que tenha capacidade para transportar 5 peças por vez. A modelagem dessa situação é feita através de um bloco Batch (considerando-se que estamos utilizando apenas blocos básicos para a nossa modelagem pois um AGV poderia ser modelado pelo bloco Transporter do Template de Transferência Avançada) da seguinte maneira:

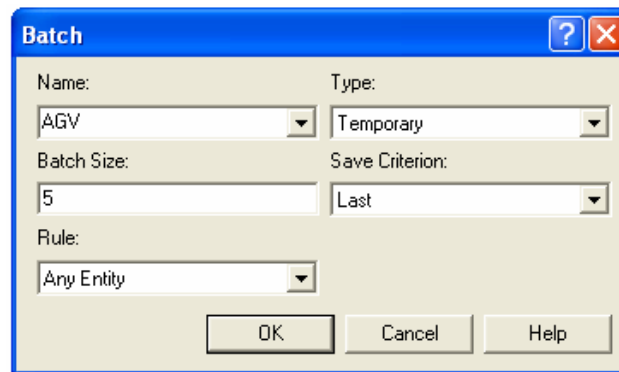


Figura 16 - Configurando de um bloco Batch temporário

Imaginemos agora então que um certo operador tem por trabalho empacotar diferentes tipos de peças, mas sempre na mesma quantidade (por exemplo, 10), onde cada tipo de peça será separado e a cada 10 peças surgirá uma “nova” entidade que será um “pacote”, os parâmetros do bloco Batch ficariam então:

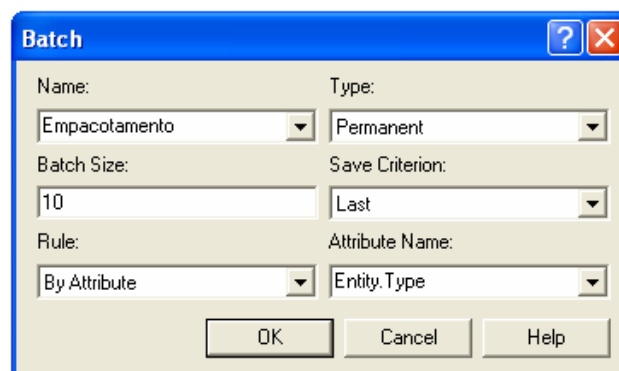


Figura 17 - Configurando um bloco batch permanente

#### 4.1.6.O Bloco Decide



Decide

Esse bloco permite a modelagem de processos de tomada de decisão no sistema. O Bloco permite decisões do tipo binária (a resposta ou é falsa ou verdadeira), ou do tipo múltipla. A escolha pode ser feita através de probabilidades (por exemplo: há 75% de probabilidade do cliente ir para a fila A e 25% de ele ir para a fila B), de atributos (por exemplo: peças do tipo A seguem para torno e peças do tipo B seguem para fresa), de expressões (por exemplo: avalia-se o tamanho da fila do processo A, se ela for maior que dois, a entidade segue para o processo B), ou variáveis do processo.

Poderíamos, por exemplo, modelar a entrada de clientes em uma video-locadora, onde dos clientes que entram, 85% acaba pegando algum filme e dirigindo-se ao caixa, enquanto 15% deles entra e sai sem levar qualquer fita ou dvd. A modelagem ficaria:

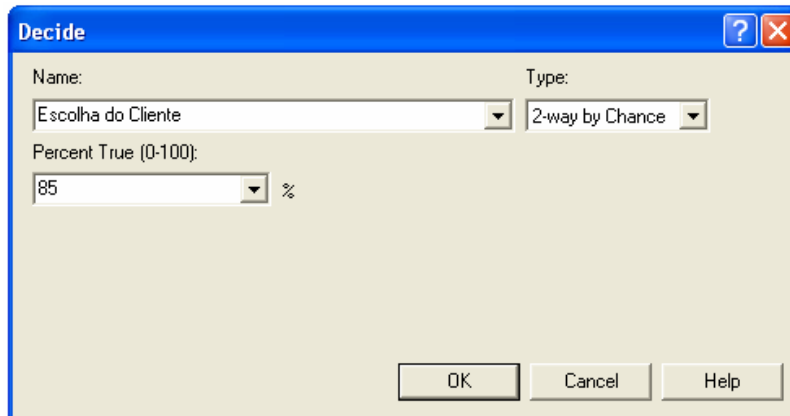


Figura 18 - Configuração de um bloco Decide baseado probabilidade

Outro exemplo que poderíamos utilizar é o de clientes que devem escolher qual fila do banco eles vão entrar (admitindo-se que o banco não utilize fila única e que só existam dois caixas operando). Faremos uma suposição de que o cliente escolha aquela em que há menos pessoas esperando, então:

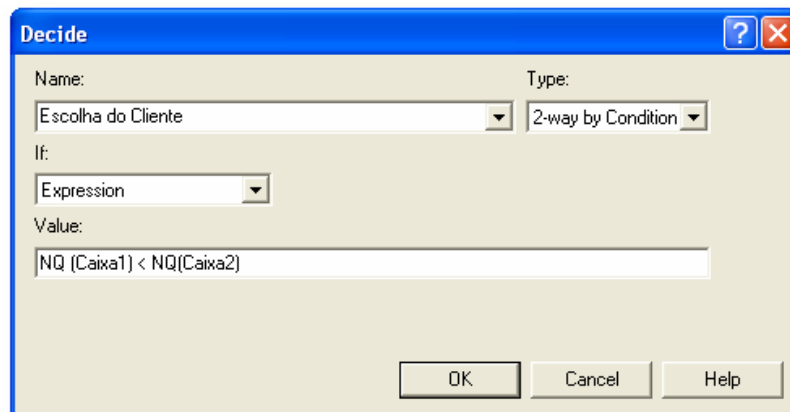


Figura 19 - Configuração de um bloco Decide baseado em condição

#### 4.1.7.O Bloco Record



Record

Esse bloco é utilizado para se conseguir informações estatísticas que vão além daquelas que são mostradas no relatório padrão que surge depois da simulação. Ele pode funcionar simplesmente como um contador, ou tirar o tempo entre as chegadas consecutivas das entidades ao bloco, ou calcular o tempo entre a chegada da entidade ao bloco e a hora em que foi atribuído um tempo qualquer a uma entidade através do bloco Assign.

Por exemplo, se eu quero saber quanto tempo um consumidor demora desde o momento em que ele entra no supermercado, até a hora em que efetivamente chega ao caixa. Logo após a entrada do mesmo, assinalamos a entidade com o tempo "atual" da simulação (TNOW) e depois de ele ter passado por todos os processos necessários, ele entra em um bloco Record e lá é calculada a diferença entre aquele tempo anterior e o novo tempo atual de simulação e grava o resultado em uma variável do tipo "Tally" que será mostrada no relatório final. O bloco Record nesse caso ficaria:

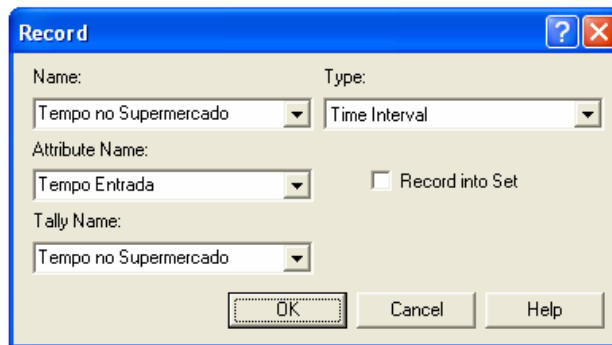


Figura 20 - Configuração de um Bloco Record para registro de Intervalo de Tempo

Outro exemplo poderia ser o caso em que queremos saber qual a taxa de chegada de entidades para um certo processo. Isso pode ser feito da seguinte maneira:

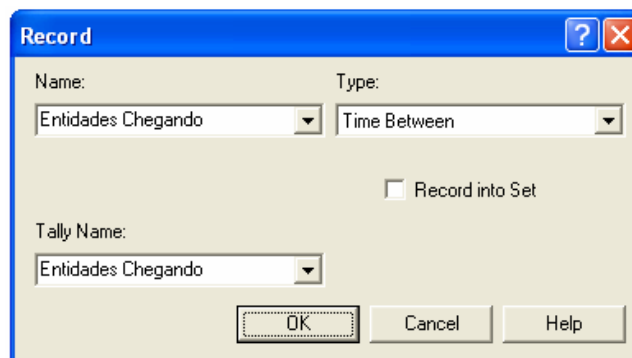


Figura 21 - Configuração de bloco Record para registro de intervalo entre chegadas

#### 4.1.8.O Bloco Separate



Separate

Esse bloco é utilizado para “desfazer” o que foi feito no bloco Batch. Caso o agrupamento tenha sido feito apenas temporariamente para que as entidades fossem processadas unidas e não individualmente, o bloco Separate os individualiza novamente de forma que eles continuem a serem processados na mesma ordem em que chegaram ao bloco de agrupamento.

Outra utilização para o mesmo bloco é o de duplicação ou “clonagem” de uma mesma entidade. Esse artifício serve em situações em que uma mesma entidade deva sofrer processamentos diferentes, realizados por recursos diferentes simultaneamente. A entidade original segue por um caminho e sua cópia por outra, depois elas são reunidas novamente através de um bloco Batch através do atributo Entity.SerialNumber que será o mesmo para as duas.

#### 4.1.9.O Bloco de Dados Entity



Entity

Nesse bloco você altera todas as informações relevantes acerca das entidades que foram criadas para o seu sistema. Atenção especial deve ser dada à coluna “Inicial Picture”, pois é aqui que será definido qual será a imagem de sua entidade percorrendo os blocos lógicos. Você pode escolher entre uma das imagens pré-existentes no Arena, ou então criar novas imagens acessando o Menu **Edit -> Entity Pictures**. É importante ressaltar que, quando uma entidade tem seu tipo renomeado ao longo do modelo é necessário utilizar um bloco Assign para alterar os atributos da mesma.

Os demais atributos das entidades se referem a parâmetros de custos usados na geração de estatísticas referentes a custos. Maiores informações podem ser obtidas na sessão que discute os relatórios.

#### 4.1.10. O Bloco Queue



Queue

O bloco *Queue* permite o controle das filas existentes no modelo. Filas são automaticamente criadas para qualquer operação que envolva espera ou acumulação de entidades. É através do bloco *Queue* que se altera a regra de ordenamento da fila podendo está ser: *FIFO*, *LIFO*, *Lowest Attribute Value* ou *Highest Attribute Value*.

#### 4.1.11. O Bloco Resource



Resource

O Bloco *Resource* permite a edição das características dos recursos utilizados pelos processos do modelo. É possível alterar a capacidade do recurso e escolher se o valor desta é fixo ou variável. Para tanto basta escolher o tipo do recurso. Caso o tipo seja *Fixed Capacity* o valor da capacidade será fixo e inserido no campo *Capacity*. Caso o tipo escolhido seja *Based on Schedule* Deverá ser montado um *Schedule* que mostre como a capacidade varia ao longo do tempo.

Além de ajustar a capacidade é possível também inserir os parâmetros para cálculo de custos associados com o recurso e ainda determinar falhas para o mesmo através do campo *Failures*. As falhas são explicadas na próxima sessão.

#### 4.1.12. O Bloco Variable



Variable

O bloco *Variable* permite determinar variáveis para o modelo bem como seus valores iniciais. As variáveis podem ser usadas em blocos de decisão, para ordenar filas, para construir expressões e o que mais a imaginação e criatividade do modelador permitir. Elas podem ser alteradas e manipuladas ao logicamente através do bloco *Assign*.

O aspecto mais importante do bloco *Variable* está na possibilidade de determinar variáveis com mais de uma dimensão e, também, determinar os valores iniciais das variáveis definidas.

#### 4.1.13. O Bloco Set



Set

O bloco *Set* permite a criação de grupos dentro do modelo. O agrupamento pode ser importante quando é possível escolher dentre um conjunto de opções. O *Set* agrupa elementos e permite que durante a simulação o ARENA esteja sempre utilizando o grupo como parâmetro e não apenas um dos elementos do mesmo. Podem ser agrupados: *Resource*, *Counter*, *Tally*, *Entity Picture* ou *Entity Type* de acordo com o tipo definido para o *Set*.

#### 4.1.14. O Bloco Schedule.



Schedule

O bloco *Schedule* permite a determinação de padrões de tempo que podem ser utilizados para simular a chegada de entidades ou a variação de capacidade de um determinado recurso.

O *Schedule* pode ser de dois tipos: *Duration* ou *Calendar*. Os *Schedules* do tipo *Duration* são editados na própria planilha de dados do *Schedule* e permitem determinar o valor da capacidade ou o número de entidades que chegam durante intervalos de tempo. *Schedules* do tipo *Calendar* são editados no editor de calendários (**Edit -> Calendar Schedules**). O campo *Scale Factor* permite manipular os valores determinados ao longo do *Schedule* e pode ser importante como parâmetro de controle. Assim, caso se determine um *Scale Factor* igual a 1.1 para valores de quantidade de entidades que entram no modelo, os valores serão multiplicados por 1.1 provocando um aumento de 10% na chegada de peças.

É importante notar alguns detalhes a respeito dos *schedules* que eles se repetem continuamente ao longo da simulação. Por exemplo, um *Schedule* que envolva um período de 8 horas, após essas 8 horas este se repetirá. O mesmo é válido para o *Schedule* do tipo calendário. É necessário que a configuração dos parâmetros de simulação estejam compatíveis com os *Schedules*.

Um outro ponto a ser ressaltado é que os *Schedules* não são determinísticos.

## 4.2. Advanced Process

Muitas das funções desse template já estão embutidas em blocos do template *Basic Process*, como o *Process* ou o *Assign*. Como o próprio nome diz, esse template deve ser utilizado principalmente para fazer uma modelagem mais detalhada e poder colher dados mais específicos ou identificar problemas mais sutis da planta.

### 4.2.1.O Bloco Delay

Este bloco atrasa uma entidade em uma quantidade de tempo especificada. Quando uma entidade entra no bloco, a expressão "*time delay expression*" é avaliada e a entidade permanece no bloco pelo tempo equivalente ao valor encontrado.

O tempo que foi "atrasado" pode ser alocado na forma de *value added*, no caso de ser um processamento ou algo que aumente o valor da entidade, *non-value added* no caso, por exemplo, de tempo gasto com resfriamento, secagem de peças ou qualquer atividade que não agregue valor, *transfer* para o tempo gasto com o transporte da entidade de um processo a outro, *wait* para qualquer tipo de tempo de espera, ou mesmo *other*, caso o atraso não se encaixe em nenhuma das categorias anteriores.

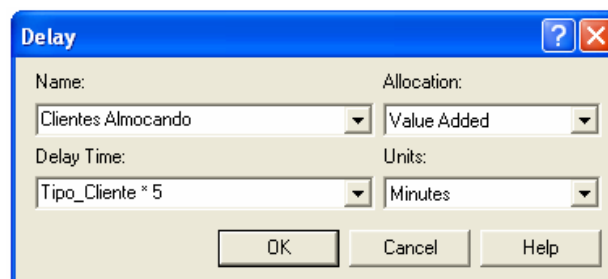


Figura 22 - Configuração do Bloco Delay: Value Added

No caso apresentado na figura 13, clientes que chegam a um restaurante sofrem um atraso que representa o tempo em que estão almoçando. Existem tipos de clientes diferentes com valores de 1 a 5 e, dependendo do tipo de cliente, há um valor para o tempo de atraso (o valor do atributo é multiplicado por 5 para obter-se o valor do atraso).

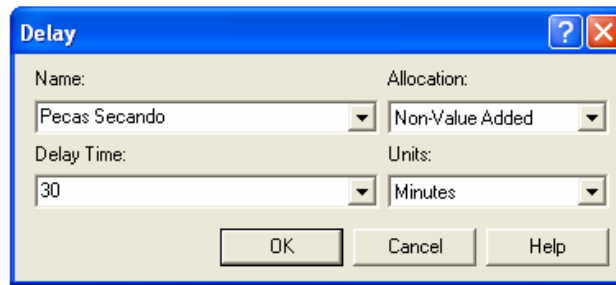


Figura 23 - Configuração do Bloco Delay: Non-Value Added

Peças que acabaram de sair do processo de pintura, seguem para uma área de secagem, onde ficam por 30 minutos “estacionadas”. O tempo é alocado como sendo “Non-Value Added” porque não agrega valor algum ao produto, apenas é um tempo necessário na produção.

#### 4.2.2.O Bloco Dropoff

O bloco *Dropoff* retira um número específico de entidades de um grupo e as envia para outro bloco, conforme especificado pela conexão gráfica. O grupo é formado quando as entidades passam por um bloco *Bath* ou por um bloco *Pickup*.

A variável NG pode ser utilizada para se avaliar o tamanho do grupo “reunido” na atual entidade e para retirar todas as entidades do grupo mantendo a entidade no sistema. Podem ser especificados atributos para as entidade que permanecem no sistema segundo uma regra específica.

A diferença entre este bloco lógico e o *Separate* é que, no segundo caso, quando é utilizada a opção *Split*, todas as entidades originais são separadas do grupo para processamentos individuais e entidade representativa do conjunto é jogada fora. O bloco *Dropoff* permite selecionar membros do grupo a serem removidos e, mesmo que todos sejam retirados, manter a entidade representativa do grupamento. As entidades representativas não podem ser retiradas do sistema antes que todas as entidades do grupo sejam separadas. Por isso caso o bloco *Dropoff* não elimine todas as entidades do grupo a entidade representativa deve passar por um bloco *Separate* antes de sair do sistema.

Caso passe uma entidade com tamanho de grupo zero, o bloco *Dropoff* irá simplesmente ignorar tal entidade e deixá-la passar pelo ponto de saída Original.

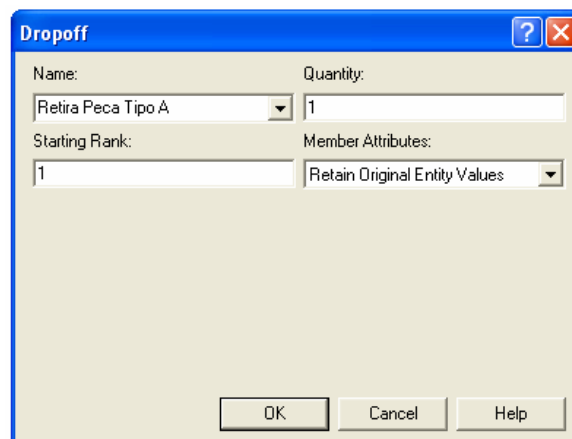


Figura 24 - Exemplo de configuração do bloco Dropoff retendo os valores originais das entidades

Em um grupo de 5 peças (representado apenas por uma entidade) a peça A é sempre a primeira do grupo e deve ser retirada em um dado instante do processamento. O grupo passa então pelo bloco *Dropoff* com as especificações



acima e a peça A é retirada. O grupo continua por um caminho e a peça A por outro. A entidade retirada manterá seus valores originais, anteriores ao seu agrupamento, como *Entity.Type*, *Entity.Station*, *Entity.CreateTime*, etc., contudo atributos temporais refletirão todos os novos tempos que tenham sido adicionados a ela quando fora parte do grupo.

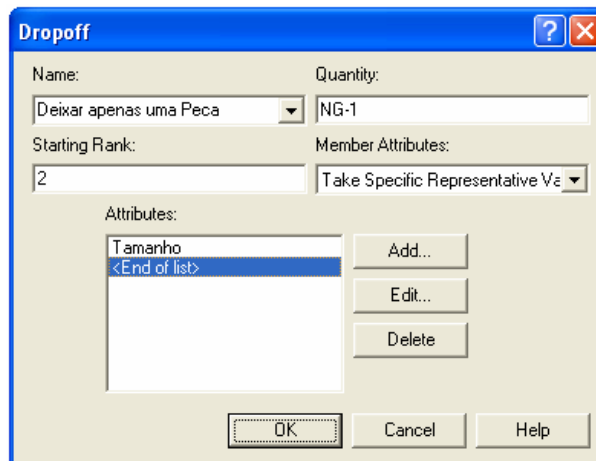


Figura 25 - Exemplo de configuração do bloco Dropoff usando valores representativos

Ao contrário do exemplo anterior, todas as peças serão retiradas do grupo, com exceção da primeira delas. Pode-se ver isso porque se começa a retirada pelo posto "2" e continua-se até "NG – 1" (Tamanho do Grupo –1). Elas reterão da entidade representativa apenas o atributo tamanho e voltarão a ter todos os outros atributos como tinham antes de serem agrupadas. Um exemplo desses seria peças que são montadas todas ao mesmo tempo como um grupo e mudam de "tamanho", quando o grupo é desfeito, a entidade tem que manter o atributo "tamanho" com esse novo valor, ao invés de manter aquele que tinha antes de ter sido agrupada.

#### 4.2.3.O Bloco Hold

Ao entrar neste bloco, a entidade será colocada numa fila, para aguardar um sinal (*Wait for signal*), esperar que uma determinada condição torne-se verdadeira (*Scan for condition*) ou ficar parada indefinidamente (deve ser posteriormente removida com o bloco Remove).

Se a entidade está aguardando um sinal, o bloco *Signal* deverá ser usado para permitir que a entidade mova-se para o próximo bloco. Se estiver aguardando que uma condição torne-se verdadeira, a entidade permanecerá no bloco (seja numa fila interna ou numa fila especificada) até que a(s) condição(ões) torne(m)-se verdadeira(s). Caso a entidade acabe permanecendo parada por tempo indeterminado (*Infinite hold*), o bloco Remove é usado para permitir que ela siga seu processamento nos blocos seguintes.

Quando uma entidade entra no bloco configurado para *Scan for Condition* e não há nenhuma fila nele, a condição é avaliada. Se verdadeira, a entidade continua seu caminho ao longo do modelo. Se falsa, ela é colocada na fila. Quando uma entidade chega ao bloco e já existem outras esperando na fila, a entidade é colocada de acordo com as condições de ranqueamento da mesma (FIFO para filas internas).

Condições com expressões numéricas são avaliadas na base do verdadeiro/falso. Zero retorna falso e qualquer valor diferente de zero retorna verdadeiro. Se a condição está baseada em um atributo, a condição muda para cada entidade que chega.

Para o caso do bloco estar configurado no modo *Wait for Signal*, cada entidade entrando no bloco recebe um código de sinal baseado em um valor específico. Através do uso de expressões com variáveis, as entidades de um mesmo bloco podem esperar por sinais diferentes.

Um bloco *Signal* pode ser usado para controlar o número de entidades que serão liberadas quando ele mandar o sinal.

The screenshot shows a 'Hold' dialog box with the following configuration:

- Name:** Espera Ordem Producao
- Type:** Wait for Signal
- Wait for Value:** 10
- Limit:** 1
- Queue Type:** Queue
- Queue Name:** Espera Ordem Producao.Que

Buttons at the bottom: OK, Cancel, Help.

Figura 26 - Exemplo de configuração do bloco Hold: Wait for Signal

Neste exemplo, as peças esperarão que um bloco Signal em algum outro ponto do modelo envie um sinal de valor "10" para que continuem o processamento. Somente uma peça será liberada a cada sinal enviado.

The screenshot shows a 'Hold' dialog box with the following configuration:

- Name:** Espera Vaga Restaurante
- Type:** Scan for Condition
- Condition:** Restaurante.WIP < 30
- Queue Type:** Internal

Buttons at the bottom: OK, Cancel, Help.

Figura 27 - Exemplo de configuração do bloco Hold: Scan for Condition

Os clientes (entidades) que atingirem esse bloco só prosseguirão se o número de clientes (entidades) sendo atendidos no restaurante (recurso) seja menor do que 30. Caso contrário, eles esperam numa fila até que outras entidades deixem o restaurante.

#### 4.2.4.O Bloco Match

O bloco *Match* "sincroniza" um número específico de entidades, fazendo-as esperar umas pelas outras em diferentes filas. É necessário haver ao menos uma entidade em cada fila para acontecer a liberação. Além disso, é possível especificar um atributo que deve ser comum a todas as entidades antes de iniciar-se o "match".

Ao chegarem ao bloco, as entidades são colocadas em filas, segundo a entrada gráfica à qual estão ligadas. Ficam então nas respectivas filas até o momento em

que exista o “match”. Neste momento, uma entidade de cada fila é liberada, e estas são então sincronizadas para partirem do bloco.

Quando ao menos uma entidade estiver presente em cada uma das filas especificadas (isso considerando o valor no campo *Number to Match*), uma sincronização ocorrerá. Se um atributo qualquer é especificado como elemento para a sincronização, essa só ocorrerá quando houver o número de entidades especificadas na fila que tenham o mesmo valor para aquele atributo.

As entidades deixam o bloco na ordem em que suas filas foram especificadas. Isso quer dizer que não saem todas exatamente “ao mesmo tempo”.

Este bloco pode ser usado em sequência ao bloco *Batch* para sincronizar números de entidades diferentes por fila. Por exemplo, sincronizar duas peças A com uma B. Agrupa-se A utilizando-se o bloco *Batch*. Sincronize e depois passe A por um *Separate*.

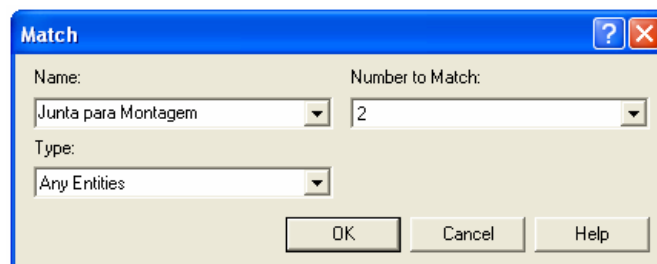


Figura 28 - Exemplo de configuração do bloco Match

Nesse exemplo, dois tipos de entidades vão chegando ao bloco *Match* por filas diferentes, quando existe uma de cada tipo, as duas são liberadas para logo seguida entrar em um bloco *Batch* e serem “montadas”, transformando-se em apenas uma entidade.

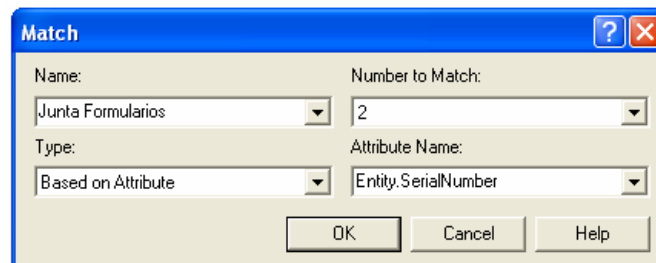


Figura 29 - Exemplo de configuração do bloco Match utilizando atributo como critério

Analogamente ao anterior, entidades vão chegando ao bloco e entrando em duas filas diferentes, provavelmente tendo vindo por caminhos diferentes do modelo. Quando duas entidades com o mesmo número de série (*Entity.SerialNumber*) estiverem, cada uma em uma fila diferente, as duas são liberadas. Isso acontece quando uma entidade foi “duplicada” em um bloco *Separate* em algum ponto anterior do sistema e cada “cópia” da entidade sofreu um processamento diferente. Nesse bloco, um dos “clones” tem que esperar pelo outro, antes de continuar.

#### 4.2.5.O Bloco Pickup

O bloco *Pickup* retira um número de entidades consecutivas de uma dada fila, começando a partir de um posto específico da mesma. As entidades retiradas são adicionadas ao final do grupo de entidades que está passando pelo bloco *Pickup*.

A quantidade a ser retirada não pode exceder o número de entidades na fila ou haverá erro de simulação. Um bloco *Decide* pode ser utilizado para examinar o

valor da variável NQ (Número de elementos na fila) de forma a evitar erros desse tipo. A variável NG pode ser utilizada para verificar o tamanho do grupo representado pela entidade que passa pelo bloco *Pickup*.

Entidades que tenham sido adicionadas a um grupo utilizando-se *Pickup* ou *Batch* podem ser removidas usando os blocos *Dropoff* e *Separate*. É importante lembrar que uma entidade representativa de um grupo não pode ser “jogada fora” até que todos os membros do grupo tenham sido removidos ou, também, haverá erro de simulação.

Se a entidade que passa pelo bloco *Pickup* não tiver passado por um bloco *Batch* anteriormente, uma entidade representativa é criada e as entidades retiradas da fila e a entidade que acionou o *Pickup* formam um novo grupo.

A principal diferença entre o bloco *Batch* e o *Pickup* é a origem das entidades que são agrupadas. No caso do *Pickup* as entidades são retiradas de uma fila que pode estar em qualquer lugar do modelo. Já o *Batch* agrupa entidades que chegam ao bloco sequencialmente.

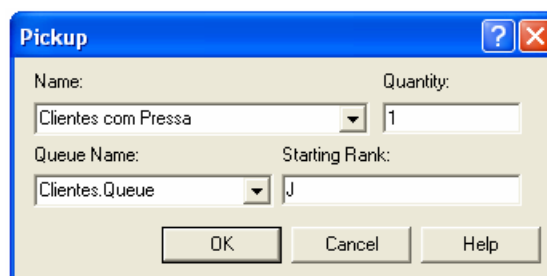


Figura 30 - Exemplo de configuração do bloco Pick

Nesse caso, um elemento da fila “Cliente” será retirado e colocado no final do grupo que estiver passando pelo bloco *Pickup*.

O elemento a ser retirado será escolhido com base na variável global J, cujo valor será conseguido através do uso de um bloco *Search* que procurará na fila por uma determinada condição (por exemplo, um atributo que indique que o cliente tem pressa) e ajustará a variável J com a posição daquele elemento da fila.

#### 4.2.6.O Bloco ReadWrite

Com este bloco é possível ler dados de um arquivo de entrada, ou do teclado, e atribuir estes valores de dados a uma lista de variáveis ou atributos. É possível ainda escrever dados em uma saída específica, como um arquivo ou a tela.

Quando uma entidade chega no bloco, o arquivo especificado é examinado para verificar se está aberto (ativo). Em caso negativo, o arquivo é automaticamente aberto. Os valores dos atributos, variáveis ou expressões (“other”) listados são lidos ou escritos de acordo com o formato especificado.

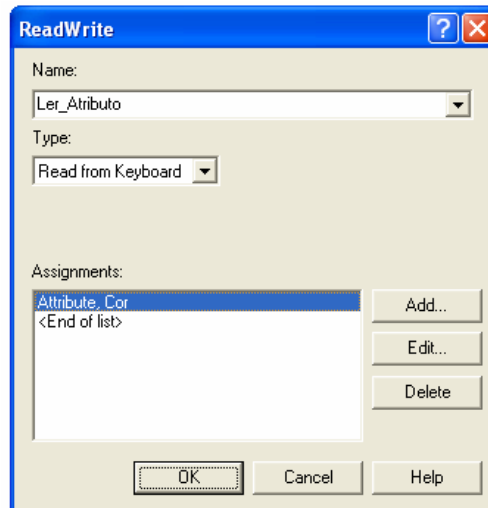


Figura 31 - Exemplo de utilização do Bloco ReadWrite

Neste exemplo, o atributo "Cor" é lido no modelo diretamente do teclado. Toda vez que uma entidade chega ao bloco *ReadWrite*, a simulação é pausada até que se receba uma resposta vinda do teclado.

#### 4.2.7.O Bloco Release

O bloco Release é usado para liberar unidades de um recurso do qual uma entidade tenha previamente se apoderado. Este bloco pode ser usado para liberar recursos individuais ou em conjunto. Para cada recurso a ser liberado, o nome e a quantidade a liberar são especificados.

Quando uma entidade entra no bloco, ela libera o controle do(s) recurso(s) especificado(s). Qualquer entidade esperando em fila(s) para aquele(s) recurso(s) ganhará (ão) controle sobre este(s) imediatamente.

A variável do sistema "NR" retorna o número atualizado de unidades ocupadas do recurso especificado. Quando uma entidade executa um bloco Release, NR é decrementado da quantidade liberada, a menos que o recurso seja imediatamente reservado por outra entidade.

Se mais unidades de um recurso específico que tenham sido previamente reservadas forem liberadas, ocorrerá erro de simulação.

A liberação múltipla de recursos será feita na ordem em que eles aparecerem no bloco Release.

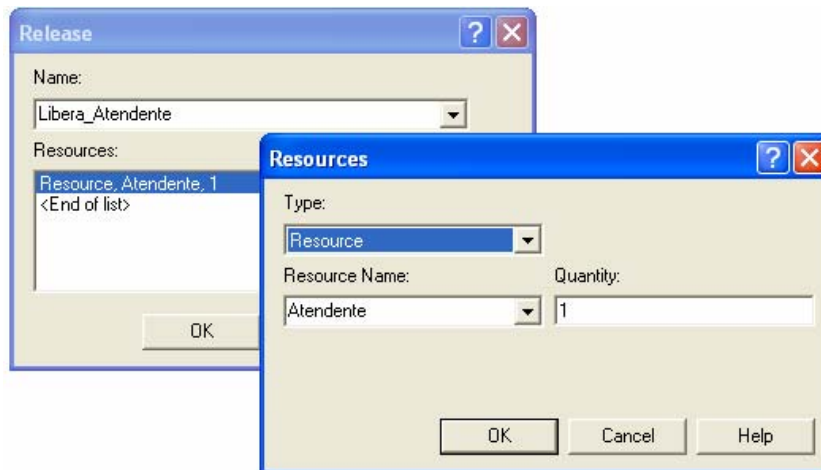


Figura 32 - Exemplo de configuração do bloco Release

No exemplo da figura 23, uma unidade do recurso “Atendente” será liberada. Se qualquer entidade estiver esperando por um “Atendente”, o recurso é imediatamente alocado para a entidade em espera com maior prioridade.

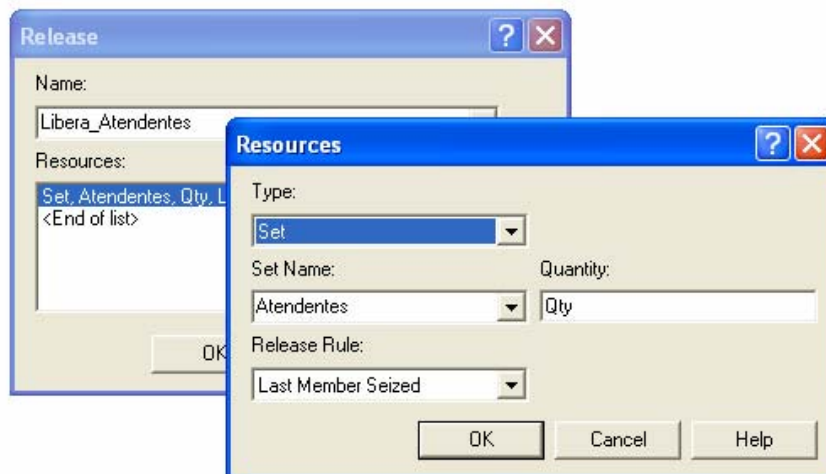


Figura 33 - Exemplo de configuração do bloco Release: Set

No caso da figura 24, o(s) membro(s) do conjunto “Atendentes” que tiver(em) sido reservado(s) por último, será(ão) liberado(s). A quantidade de unidades liberadas será dada pelo valor do atributo Qty da entidade que estiver passando pelo bloco *Release*.

#### 4.2.8.O Bloco Remove

Este bloco retira uma única entidade de uma posição específica em uma fila e a envia para um bloco designado. Quando uma entidade chega ao bloco *Remove*, uma entidade da fila especificada é retirada e enviada ao bloco especificado. O posto (rank) da entidade significa a posição da mesma na fila.

A entidade que causou a retirada segue para o bloco seguinte e é processada antes da entidade proveniente da fila.

A última entidade na fila pode ser removida especificando-se o posto da mesma como a palavra-chave NQ ou como a variável NQ. (Nome da Fila).

Antes de enviar uma entidade para o bloco *Remove*, verifique se a entidade a ser removida realmente se encontra na fila através do bloco *Search*. Esse bloco seta a variável global J como o posto da entidade desejada, ou com 0 (zero) se

nenhuma delas têm às características desejadas. Ocorrerá erro de simulação se o bloco *Remove* tentar remover uma entidade com posto maior que o tamanho da fila.

Não há como remover entidades que estejam em filas internas.

#### 4.2.9.O Bloco Seize

O bloco *Seize* aloca unidades de um ou mais recursos para uma entidade. Pode ser usado para alocar unidades de um recurso particular, um membro de um conjunto de recursos, ou um recurso definido por um método alternativo, como um atributo ou uma expressão.

Quando uma entidade entra no bloco, ela aguarda em uma fila (se especificado) até todos os recursos solicitados estarem simultaneamente disponíveis. Também deve ser especificado o tipo de alocação para o uso do recurso.

Entidades que reservam o recurso com valores de prioridades mais baixos têm prioridade sobre aquelas com valores mais altos. Números negativos são transformados em 0 (zero). Se várias entidades tentarem reservar o mesmo recurso e possuírem valores de prioridade idênticos, a entidade com o maior tempo de espera receberá o recurso.

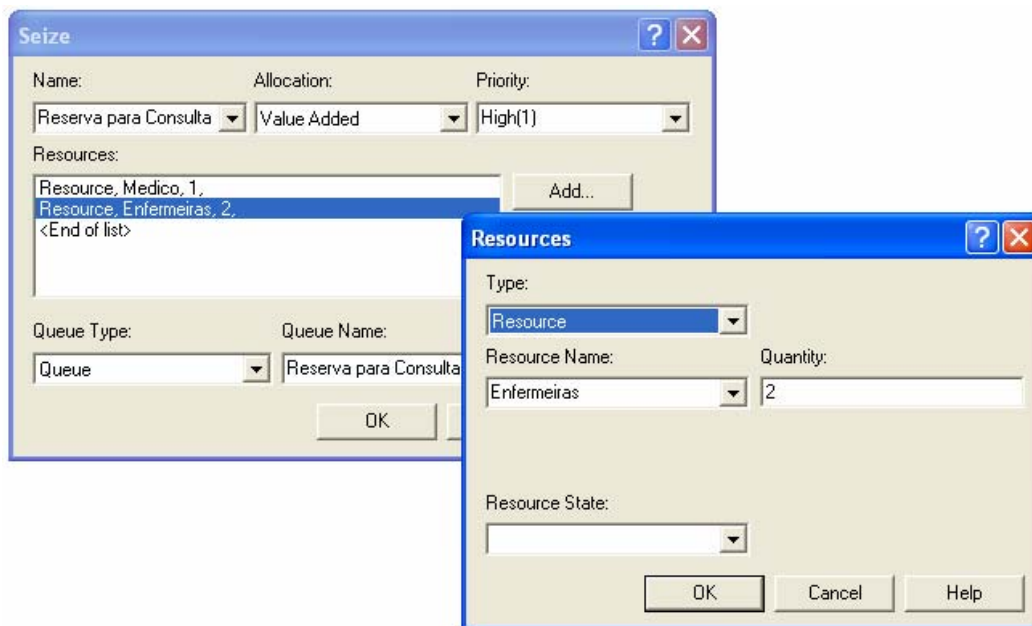


Figura 34 - Exemplo 1 de configuração do bloco Seize

No exemplo 1, as entidades esperam na fila até que um médico e uma enfermeira estejam disponíveis para a consulta. As entidades que estão na fila esperando têm prioridade sobre quaisquer outras entidades que estejam esperando por esses mesmos recursos e que tenham prioridade de valor maior do que 1. Quando os recursos são reservados, o estado deles será “ocupado”, uma vez que não há nenhum outro estado especificado.

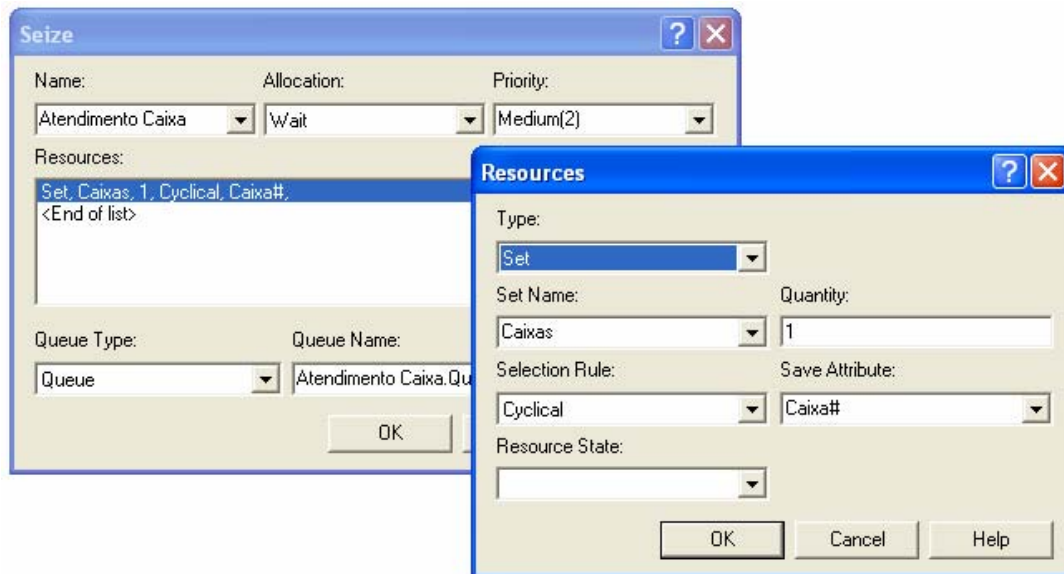


Figura 35 - Exemplo 2 de configuração do bloco Seize

No exemplo 2, a entidade deseja reservar um dos caixas que o banco possua para ser atendido. Se mais de um deles estiver disponível, eles serão escolhidos um por vez, de maneira cíclica, pelas entidades que forem chegando ao bloco. A informação sobre qual dos caixas foi reservado é armazenada no atributo da entidade de nome "Caixa#". Essa informação será útil na hora de liberar o recurso.

#### 4.2.10. O Bloco Search

Este bloco realiza uma busca em uma fila, um agrupamento (batch) ou uma expressão para encontrar o posto (rank) de uma entidade (para filas ou agrupamentos) ou o valor da variável global J que satisfaça a condição de busca especificada. Quando da busca em filas ou agrupamentos, o valor da variável global J recebe a posição (rank) da primeira entidade que satisfaça a condição de busca, ou o valor 0 se a condição de busca não é satisfeita. Analogamente, quando da busca em expressões, J recebe o primeiro valor que satisfaça a condição especificada, ou então 0 se nenhum valor de J dentro do intervalo especificado satisfaz a condição de busca.

Quando uma entidade chega no bloco *Search*, J recebe o valor de início (starting index) e a condição de busca é avaliada. Se satisfeita, a busca é finalizada e o valor de J é mantido. Caso contrário, o valor de J é incrementado (ou decrementado) e a condição é reavaliada. Esse processo é repetido até satisfazer-se a condição de busca ou o valor final ser atingido.

Caso não existam entidades na fila ou grupo, ou não seja encontrada a condição de busca, J recebe valor 0.



Figura 36 - Exemplo 1 de configuração do bloco Search

No exemplo 1, o bloco *Search* é configurado para fazer uma busca na fila do empacotamento de forma a encontrar uma entidade cujo atributo peso tenha o menor valor.

Figura 37 - Exemplo 2 de configuração do bloco Search

No exemplo 2, o bloco *Search* procura no grupo que passa por ele a posição de uma entidade com o valor do atributo "*StatusTrab*" maior que 3. A busca é feita de trás para frente e caso não seja encontrada nenhuma entidade com essas características, *J* é setado em zero.

#### 4.2.11. O Bloco Signal

Este bloco envia um sinal para todos os blocos do tipo *Hold* existentes no modelo que estejam configurados em "*Wait for Signal*", liberando o maior número especificado de entidades.

Quando uma entidade entra no bloco, "*SignalValue*" é avaliado e o código do sinal é enviado. Neste momento, entidades presentes em blocos *Hold* que estavam esperando pelo mesmo sinal são retiradas de suas filas. A entidade que originou o sinal continua a ser processada, até que encontre uma espera, uma fila, ou seja, retirada do sistema.

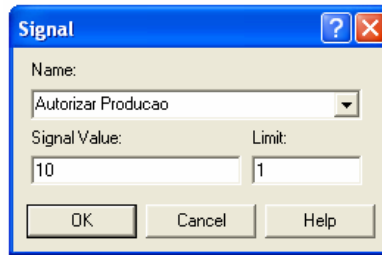


Figura 38 - Exemplo de configuração do bloco Signal

No exemplo da figura 29, quando uma entidade passa pelo bloco *Signal*, será liberada a primeira entidade que esteja esperando em algum bloco *Hold* no sistema o sinal "10".

#### 4.2.12. O Bloco Store

O bloco *Store* adiciona uma entidade a um local de armazenamento. O bloco *Unstore* pode ser usado posteriormente para retirar de lá a entidade.

Quando uma entidade chega no bloco, o armazenamento em questão é incrementado, e a entidade segue imediatamente para o bloco seguinte.

"Storages" são úteis para mostrar animações de uma entidade enquanto esta é processada por outros blocos. Além disso, podem ser colhidas estatísticas sobre o número de entidades armazenadas.

A variável *NSTO* contém o número de entidades em um armazém especificado. Se essa variável crescer indefinidamente durante a simulação, é possível que não tenha sido colocado no modelo um bloco *Unstore* e as entidades não estejam sendo retiradas do armazém.

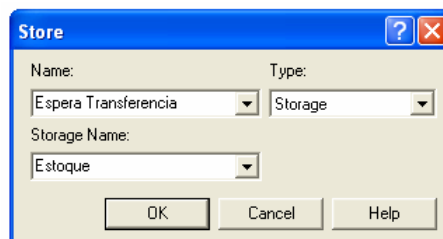


Figura 39 - Exemplo de configuração do bloco Store

Na figura 30 vemos um exemplo de utilização do *Store* onde as entidades que chegam a esse bloco *Store* são adicionadas ao armazém de nome "Estoque". Se uma animação de armazém é criada com esse mesmo nome no modelo, a figura da entidade será adicionada ao armazém e permanecerá lá até que no fluxo lógico ela passe por um *Unstore* ou até que outra atividade específica de animação como uma transferência a tire de lá.

#### 4.2.13. O Bloco Unstore

Este bloco retira uma entidade de um local de armazenamento. Quando uma entidade chega no bloco, o armazenamento em questão é decrementado e a entidade segue imediatamente para o próximo bloco.

A entidade que entra nesse bloco deve ter previamente passado por um bloco *Store*, caso isso não aconteça, um aviso aparecerá na tela. E as entidades devem ter sido removidas de todos os armazéns antes que possam ser "jogadas fora" (através de um bloco *Dispose*).

#### 4.2.14. Adjust Variable

O bloco *Adjust Variable* permite ajustar o valor de uma variável segundo um valor padrão a uma taxa especificada. Esse bloco pode ser usado com a opção *Rotate By Expression* em figuras de recursos e globais para apresentar uma animação suave de rotação de imagens (por exemplo, rotacionar a imagem de um recurso 180 graus a uma taxa de 5 graus por segundo). O bloco também pode ser usado para aproximar/animar um aumento contínuo de uma variável ao longo do tempo.

Quando uma entidade chega ao bloco, a variável escolhida é ajustada de para o valor de acordo com a taxa especificada. A entidade permanece no bloco até que o ajuste tenha sido concluído.

O *Update Interval* especifica um intervalo de tempo que será observado entre as atualização da variável. No caso da animação de imagens, um intervalo de tempo menor produz animação mais suave e estatísticas mais precisas referentes a variável. A definição de um intervalo menor produz maior velocidade na execução da simulação.

#### 4.2.15. Advanced Set

O bloco *Advanced Set* permite especificar conjuntos de “*queues*”, *storages*, dentre outros. Um set, como já visto anteriormente define um grupo de elementos semelhantes que podem ser referenciados através de uma nome comum (o nome do set) e um index dentro do grupo.

Sets de “*queues*” também podem ser especificados em blocos *Seize*, ou outros blocos que tratam a manipulação de recursos. *Storage* sets podem ser usados nos blocos *Store* e *Unstore*. Outros sets podem incluir sets de atributos ou outros elementos mistos de acordo com o interesse do modelador.

#### 4.2.16. Expression

O bloco *Expression* permite definir expressões e seus valores associados. Tais expressões podem ser referenciadas no modelo através do seu nome. Existe ainda a possibilidade de definir expressões como arrays de uma ou duas dimensões.

#### 4.2.17. Failure

O bloco *Failure* é utilizado juntamente com o bloco *Resource* (campo *failure*). Quando ocorre a falha o recurso tem seu estado como *failed* (em falha). O bloco *Failure* foi elaborado para ser usado com recursos de capacidade unitária ou com recursos de capacidade múltipla cujos recursos individuais falham todos ao mesmo tempo.

#### 4.2.18. File

O bloco *File* deve ser incluído sempre que se deseja acessar arquivos externos usando o bloco *ReadWrite*. Esse bloco identifica o nome do arquivo e define o método de acesso, formatação e características operacionais do arquivo.

#### 4.2.19. State Set

O bloco *State Set* é usado para definir estados para recursos. Os estados podem ser associados com um autoestado, ou podem ser novos estados. O bloco *Resource* do template *Basic Process* referencia o *stateset* que o recurso usará.

#### 4.2.20. Statistic

O bloco *Statistic* é usado para definir estatísticas adicionais a ser coletadas durante a simulação, bem como para especificar arquivos de dados de saída. Estatísticas elementares são automaticamente geradas (por exemplo média, mínimo, etc), através do bloco *Statistic* um arquivo de saída pode ser criado e guardado o histórico da observação que será escrito no arquivo de saída.

Os tipos de estatísticas que podem ser definidas através do bloco *Statistic* são: *time-persistent, tallies, count-based, outputs, and frequency-based*.

#### 4.2.21. Storage

O bloco *Storage* define o nome de uma espécie de “armazém”. “*Sotrages*” são automaticamente criados por qualquer bloco que face referência de forma que o bloco *Storage* raramente precisa ser usado. Num entanto o uso será necessário quando um “armazém” é especificado usando atributos ou expressões.

### 4.3. Advanced Transfer

#### 4.3.1.O Bloco Enter

O bloco *Enter* é tipicamente o primeiro bloco de um conjunto usado para definir uma ou mais etapas de processamento. Ele define uma estação (ou um conjunto de estações) correspondendo ao espaço físico ou lógico onde ocorrerá o processamento.

Uma entidade pode mover-se do bloco anterior para o bloco *Enter* de duas maneiras: transferindo para a estação associada com o bloco ou através de uma conexão gráfica.

Quando uma entidade chega ao bloco *Enter*, um atraso de descarregamento pode ser selecionado e algum equipamento de transferência que tenha sido utilizado para transferir a entidade ao bloco *Enter* pode ser então liberado.

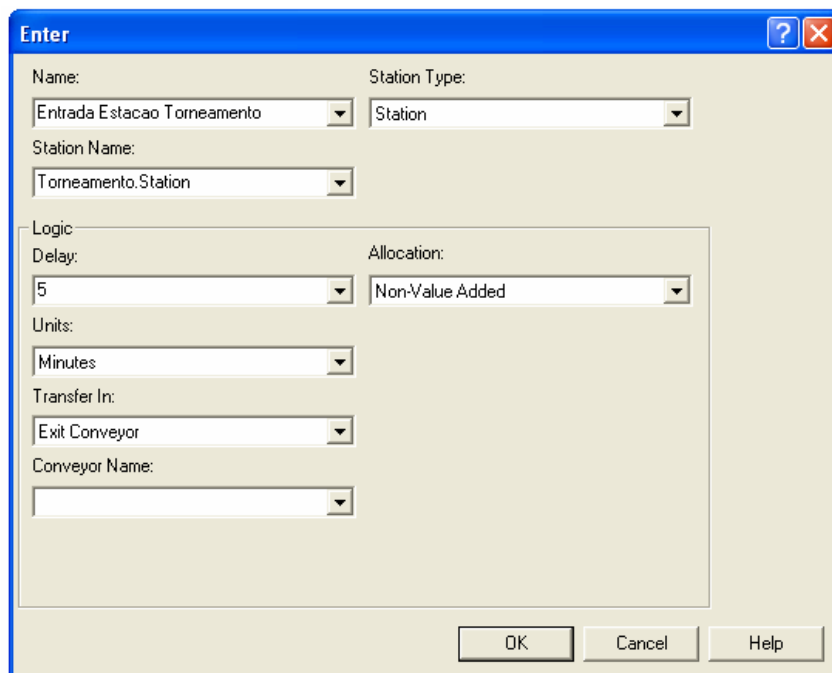


Figura 40 - Exemplo de configuração do bloco Enter

No exemplo apresentado na figura 31, peças ou entidades chegam a uma estação destino chamada "Torneamento". Cada entidade que chega entra em um delay de cinco minutos para ser descarregada do transportador que a traz. Depois que o descarregamento tiver sido completado, ela libera o seu lugar no transportador. Não é necessário identificar qual o nome do transportador deve ser liberado, pois já fica subentendido que é aquele em que a peça se encontra.

Durante o descarregamento de cinco minutos, a esteira (transp.) ficará completamente parada (no caso de transp. não acumulativos) ou a entidade causará um bloqueamento local na proximidade da entrada da estação destino (no caso de transp. acumulativos).

#### 4.3.2.O Bloco Leave

Este bloco é usado para transferir uma entidade para uma estação ou para outro bloco. A transferência pode ocorrer de duas maneiras, graficamente ou referenciando-se a estação, roteamento, condutor ou transportador.

Quando uma entidade chega a este bloco, ela pode ainda aguardar para obter acesso ao equipamento de transferência a ser utilizado, (recurso, transportador ou condutor). Quando este é obtido, a entidade pode sofrer um atraso de carregamento, sendo em seguida transferida do bloco até seu destino posterior.

Quando requisitando um transportador, ele deve ser definido em um bloco *Transporter* e um bloco *Distance* deve ser utilizado para definir o caminho que o transportador seguirá entre as estações.

Quando acessando um condutor, esse deve ser definido em um bloco *Conveyor* e um outro bloco *Segment* deve ser utilizado para definir como será o fluxo de entidades sobre ele.

O bloco *Leave* é normalmente um bloco de "fechamento" de uma região que define uma área de trabalho. É o último em uma série que se inicia com, por exemplo, um bloco *Enter*, seguido por um *Seize*, um *Delay*, um *Release*, ou simplesmente um *Process* e finalmente termina aquela "região" com um bloco *Leave*.

The image shows a software configuration window titled "Leave". It contains the following fields and options:

- Name:** Saindo da Sala de Espera
- Allocation:** Non-Value Added
- Delay:** UNIF (15,22)
- Units:** Seconds
- Logic:** Transfer Out: None
- Connect Type:** Route
- Move Time:** 5
- Units:** Second
- Station Type:** Station
- Station Name:** Consultorio

Buttons at the bottom: OK, Cancel, Help.

Figura 41 - Configuração do bloco Leave sem ação de transporte

No exemplo acima (figura 32), não é necessário nenhum dispositivo para mover as entidades do bloco *Leave*. As entidades sofrem um tempo de preparação ou de carregamento antes de deixar a estação baseado em uma distribuição uniforme com um mínimo de 15 segundos e um máximo de 22 segundos (poderia simbolizar algum formulário que o paciente tivesse que assinar ou alguma outra coisa que o "atrasa"). Depois disso, a entidade, que é uma pessoa no caso, segue por conta própria para a próxima estação "Consultório", levando para isso 45 segundos.

Já no próximo exemplo (figura 33), é necessário um AGV para levar as peças de uma estação à outra. Um entre os AGVs disponíveis é requisitado (e seu "código" é armazenado no atributo "Numero\_AGV" para uso posterior) e move-se até a estação em que a peça encontra-se. Se não houver nenhuma unidade do transportador disponível no momento, a entidade esperará em uma fila interna. Uma vez que o AGV chegar à estação de origem, haverá um tempo de "carregamento" de três minutos. Quando esse processo terminar, a entidade será transportada para a próxima estação "Fresamento" segundo especificações feitas nos blocos *Transporter* e *Distance*.

Figura 42 - Configuração do do bloco *Leave* com ação de transporte

#### 4.3.3.O Bloco *PickStation*

Com este bloco é possível que uma entidade selecione uma estação dentre múltiplas estações especificadas, segundo uma lógica de decisão definida no bloco.

O processo de seleção é baseado num valor mínimo ou máximo de uma ou mais variáveis e expressões.

Quando for utilizado um transportador ou uma esteira para a transferência de uma estação a outra, é necessário que a entidade anteriormente tenha reservado para si um desses tipos de movimentadores. Blocos como o *Request* (transportadores) e *Access* (condutores) devem ter sido acionados em posição anterior no fluxo com relação ao bloco *Pickstation*.

#### 4.3.4.O Bloco Route

Este bloco transfere uma entidade para uma estação específica, ou para a próxima estação em uma visitação seqüencial de estações definida para a entidade. Um atraso para a transferência pode ser definido.

Quando uma entidade entra no bloco, seu atributo *Entity.Station* é alterado para a estação de destino. A entidade é então enviada ao seu destino, segundo o tempo especificado.

A variável *NE* pode ser usada para descobrir o número de entidades que estão dirigindo-se para uma determinada estação seja através do bloco *Route*, *Transport* ou *Convey*.

Figura 43 - Exemplo de configuração do bloco Route 1

Nesse exemplo (figura 34), a entidade que entra no bloco *Route* segue para a estação “Armazem” em um tempo que varia de entidade para entidade segundo uma exponencial de média igual a cinco. Na animação, a própria entidade aparecerá seguindo pelo caminho traçado para ela até o armazém.

Figura 44 - Exemplo de configuração do bloco Route 2

Já no segundo exemplo (figura 35), cada entidade demorará um certo tempo para alcançar o armazém seguindo um valor específico que esteja indicado no seu atributo “Tempo\_Entidade”.

#### 4.3.5.O Bloco Station

O bloco *Station* define uma estação (ou um conjunto de estações) que corresponde a um local físico ou lógico onde ocorre o processamento. Se o bloco define um conjunto de estações (*Station Set*), está na verdade definindo múltiplos locais de processamento.

A entidade vai movimentar-se diretamente de um dos blocos onde a transferência é iniciada (*Route*, *Transporter* ou *Convey*) para o próximo bloco *Station* não importa em que lugar da tela do computador esse esteja. Não há conexão “física” entre as “ilhas” formadas pelas estações de trabalho. Elas

conectam-se através de transferência, ou seja, a passagem entre uma é mostrada na animação, mas não no fluxo lógico.

#### 4.3.6.O Bloco Access

Este bloco aloca uma ou mais células de um condutor a uma entidade para a movimentação de uma estação para outra. Após a posse das células pela entidade, essa pode ser conduzida para a próxima estação.

Ao chegar neste bloco, a entidade aguardará até que o número apropriado de células contíguas do condutor estejam vazias e alinhadas com a estação.

O atributo *Entity.Station* da entidade deve estar com um valor válido quando a entidade tentar acessar um condutor. Esse é um erro muito comum que pode acontecer e normalmente é causado por um bloco *Separate*. Explicando melhor: quando um grupo é formado em um bloco *Batch*, as entidades perdem seus atributos individuais e ficam com os da entidade representativa, ou seja, se ela mudar de estação ainda como membro de um grupo, seu *Entity.Station* não irá mudar. Quando o grupo passar pelo *Separate* e a opção *Retain all original values* estiver acionada (é a opção default), a entidade voltará a ter como valor em seu *Entity.Station* aquele que tinha antes de tornar-se membro do grupo. Dessa maneira, quando ela tentar acessar um condutor, sua entidade de “origem” não corresponderá àquela definida no bloco.

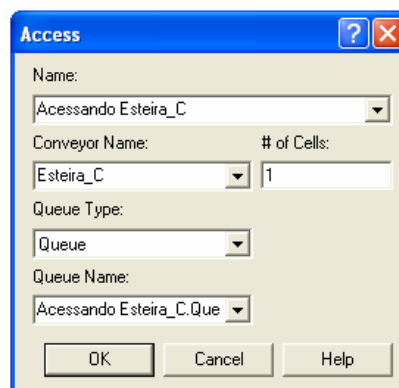


Figura 45 - Exemplo de configuração do bloco Access 1

Nesse exemplo (figura 36), a entidade deseja movimentar-se no condutor “Esteira\_C”. Somente uma célula do condutor é necessária para a entidade movimentar-se. Quando uma entidade entrar no bloco, ela espera na fila “Acessando Esteira\_C.Queue” até que uma célula esteja disponível.

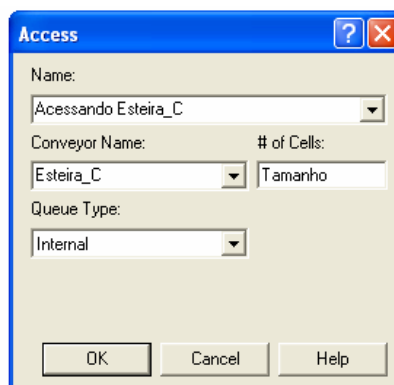


Figura 46 - Exemplo de configuração do bloco Access 2



Nesse caso, o número de células que deverão estar disponíveis para que a entidade acesse o condutor dependerá do valor do atributo "Tamanho" de cada uma das entidades.

#### 4.3.7.O Bloco Convey:

Este bloco é responsável pelo transporte de uma entidade de uma estação a outra através de um condutor.

O tempo gasto no transporte da entidade de uma estação à próxima é baseado na velocidade do condutor (especificada no bloco *Conveyor*) e na distância entre as estações (especificada no bloco *Segment*).

Quando uma entidade entra neste bloco, seu atributo *Entity.Station* é definido como sendo a estação de destino.

Se o tipo da estação de destino é definido como "Sequential", a próxima estação é determinada pelos atributos *Entity.Sequence* e *Entity.Jobstep* da entidade.

É necessário que a entidade tenha acessado células no condutor, através do bloco *Access*, antes de ser conduzida a sua estação de destino ou, sendo mais específico, antes de entrar no bloco *Convey*.

Se o nome do condutor a ser utilizado não for especificado, fica subentendido que a entidade deve ser transportada através do último condutor que ela tenha acessado (através do bloco *Access*).

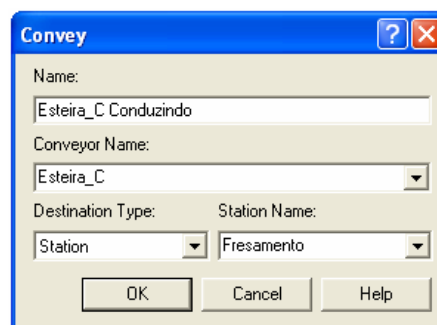


Figura 47 - Exemplo de configuração do bloco Convey

Por exemplo, como podemos ver na figura 38, a peça ou entidade será conduzida da estação de origem (que pode ser, como nos exemplos anteriores, a estação "Torneamento") até a estação destino "Fresamento" através do condutor "Esteira\_C". A entidade deve ter previamente acessado esse condutor em uma estação de origem que esteja informada em seu atributo *Entity.Station* e definida no bloco *Segment*.

#### 4.3.8.O Bloco Exit

O bloco *Exit* libera as células ocupadas pela entidade em um condutor. Se outra entidade está aguardando células disponíveis na mesma estação onde esta liberação ocorre, ela acessará então o condutor.

Parece óbvio, mas não é incomum que seja fonte de erros, que a entidade deva ter previamente acessado um condutor através de um bloco *Access* para que ela possa "sair" dele através de um bloco *Exit*.

Caso o nome do condutor não seja especificado, a entidade sairá daquele que tiver sido acessado mais recente pela mesma.

Uma entidade que tenha acessado células em um condutor não precisa necessariamente ser conduzida por ele. Ela pode simplesmente acessar as células, sofrer um atraso e depois liberar as mesmas. Isso pode servir para simular, por

exemplo, um bloqueio ou alguma coisa que faça parar o condutor, mas nesses casos é melhor usar os blocos específicos para falhas.

Como pode ser visto pelo exemplo (figura 39), não há segredo algum nesse bloco, é como um *Dispose*. Afinal a entidade simplesmente libera aquilo que estava usando e não há necessidade de especificar-se coisa alguma, nem mesmo o número de células, pois serão aquelas estivessem sendo ocupadas pela entidade (a não ser algum caso muito específico).

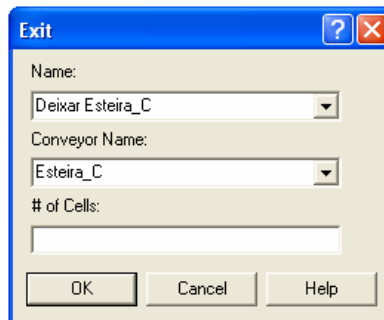


Figura 48 - Exemplo de configuração de um bloco Exit

#### 4.3.9.O Bloco Start

Este bloco retira condutor do estado inativo. No momento do *Start*, o estado pode alterar-se de inativo (3) para em espera (*idle*) (0), ativo (*moving*) (1), ou mesmo bloqueado (*blocked*) (2). Nesse momento é ainda possível alterar a velocidade do condutor.

As entidades podem continuar no condutor enquanto ele estiver inativo. Elas estarão nas mesmas posições que antes depois que o condutor voltar a se movimentar.

Aplicar *Start* em um condutor que já se encontra operante não terá qualquer efeito sobre o mesmo.

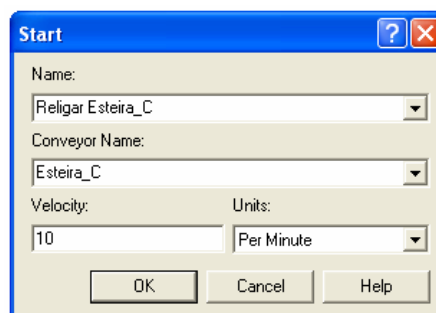


Figura 49 - Exemplo de configuração de um bloco Start

No exemplo acima (figura 40), o condutor “Esteira\_C” é religado. A sua velocidade a partir de então será de 10 unidades de tamanho por minuto. Este valor poderá ser alterado em qualquer momento da simulação através do uso da variável VC.

#### 4.3.10. O Bloco Stop

Este bloco define o estado de um condutor como inativo. Quando uma entidade entra no bloco, o condutor indicado pára imediatamente, independentemente do tipo de condutor ou número de entidades por ele sendo conduzidas.

No momento em que o condutor é desativado, sua variável de status *ICS* é alterada para inativo (*inactive*) ou valor 3.

Entidades que estejam sobre o condutor no momento em que ele parar, permanecerão nas suas respectivas posições até que alguma entidade atravesse um bloco *Start* que reative o condutor.

Parar (*Stop*) um condutor cujo corrente estado é inativo não terá qualquer efeito sobre o mesmo.

Alterar a velocidade de um condutor, representada pela variável *VC*, para zero não é a mesma coisa que pará-lo ou deixá-lo inativo. Quando a velocidade é levada a zero, o status do condutor permanece o mesmo, ainda que ele não se mova.

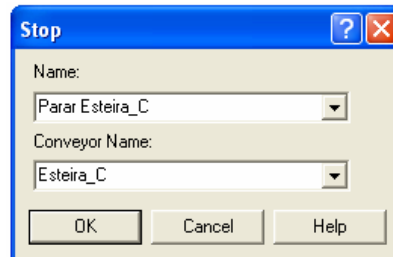


Figura 50 - Exemplo de configuração de um bloco Stop

Assim como outros blocos já mencionados, o *Stop* é um bloco bastante simples e objetivo. Sua ação é pontual e refere-se a um elemento específico do sistema, portanto não há muito a ser especificado além do próprio condutor que deverá tornar-se inativo.

#### 4.3.11. O Bloco Activate

Para transportadores inicialmente definidos como inativos ou para os que foram previamente "desativados" por alguma entidade, deve-se utilizar este bloco para realizar sua ativação. A unidade do transportador permanecerá na estação onde ele foi parado até que uma entidade solicite seu deslocamento.

No momento do ativamento do transportador, a variável *IT* é alterada do valor 2 (*inactive*) para 0 (*idle*) ou para 1 (*busy*) caso existam outras entidade esperando para utilizar o transportador.

Esse bloco é o dual dos transportadores para o *Start* dos condutores.

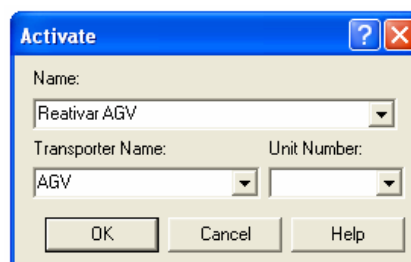


Figura 51 - Exemplo de configuração de um bloco Activate

No exemplo da figura 42, o transportador "AGV" é reativado depois de ter sido previamente inativado em alguma outra parte do modelo. Como o campo *Unit Number* não foi especificado, subentende-se que seja a unidade de código "1" ou mesmo que só exista uma unidade desse transportador no sistema.

Diferentemente do caso anterior, na figura 43, a unidade a ser reativada é aquela que estiver associada ao atributo *Numero\_AGV*. Em algum bloco anterior do tipo *Halt*, uma unidade AGV deve ter sido especificada e seu valor inserido nesse mesmo atributo.

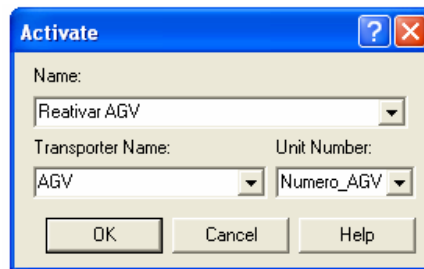


Figura 52 - Exemplo de configuração de um bloco Activate 2

#### 4.3.12. O Bloco Allocate

Este bloco designa um transportador a uma entidade sem que essa necessariamente desloque-se junto com ele e sem que o transportador mova-se até onde está a entidade. Ela então pode conduzi-lo até um ponto específico ou pará-lo.

Pode ser alocada uma unidade específica de um transportador, ou ainda utilizar-se uma regra de seleção para determinar qual dos transportadores atenderá a mesma.

Esse bloco pode ser utilizado principalmente quando se deseja enviar um transportador para um determinado local sem que a entidade que fará o envio seja "carregada" pelo transportador.

A entidade que entra no bloco deve ter de qualquer maneira um valor válido alocado no seu atributo *Entity.Station*, pois se o método de seleção da unidade do transportador a ser alocada for o de *Largest Distance* ou o de *Smallest Distance*, essa informação poderá ser utilizada para se fazer os cálculos da distância. Obviamente que as regras de seleção só são válidas quando há mais de uma unidade disponível do transportador para ser alocada.

Se duas entidades de prioridade iguais estiverem tentando alocar um transportador em blocos diferentes, aquela que estiver mais próxima será a escolhida.

O estado das variáveis do transportador, *NT* e *IT*, altera-se quando o transportador é alocado para alguma entidade. *NT* representa o número de entidades em estado "ocupado" (*busy*) no conjunto de transportadores e é incrementada de 1 quando alocada. *IT* indica o estado de uma unidade em particular do transportador e muda para "ocupado" (*busy*), ou "1".

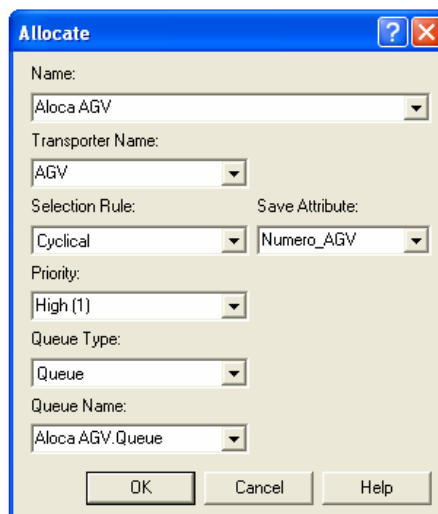


Figura 53 - Exemplo de configuração do bloco Allocate

No exemplo acima (figura 44), a entidade que entra no bloco *Allocate* tenta conseguir o controle sobre um das unidades do transportador "AGV" e conseguirá daquela que estiver mais próxima a esteja também disponível. O código ou número da unidade que estiver sendo alocada será "gravado" no atributo "Numero\_AGV" da entidade. Se não houver nenhuma entidade disponível naquele momento, a entidade espera na fila "Aloca AGV.Queue".

#### 4.3.13. O Bloco Free

Este bloco *Free* libera o transportador mais recentemente utilizado pela entidade.

Se não existem entidades aguardando a liberação do transportador, este ficará ocioso na estação da entidade onde ocorreu a liberação, a menos que esteja especificado diferentemente no bloco "*Transporter*".

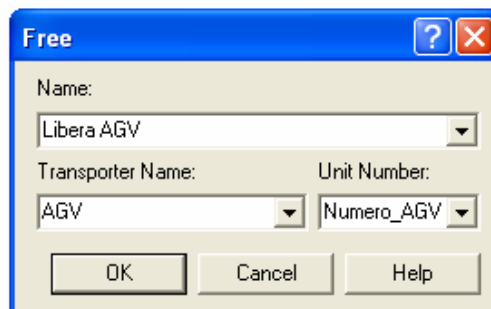


Figura 54 - Exemplo de configuração do bloco Free

Outro bloco de uso extremamente simples, seguindo o exemplo do bloco anterior (*Allocate*), aqui a entidade estaria liberando a mesma unidade do transportador "AGV" que havia alocado anteriormente, pois o código da unidade estaria gravado no atributo "Numero\_AGV".

#### 4.3.14. O Bloco Halt

Este bloco altera o status de uma unidade do transportador para "inativo". Se ele está ocupado, seu estado será "ocupado" e "inativo", até que a entidade que o controla libere a unidade. Se ele está ocioso no momento em que uma entidade o define como inativo, ele se torna inativo imediatamente.

O transportador neste estado não poderá dar acesso a nenhuma entidade antes de ser "ativado" novamente.

O bloco *Halt* é o equivalente para transportadores do bloco *Stop* utilizado para desativar condutores.

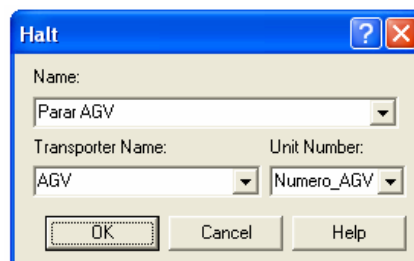


Figura 55 - Exemplo de configuração do bloco Halt

Uma vez que a entidade tenha controle sobre alguma unidade do transportador "AGV", no caso aquela que esteja identificada no atributo "Numero\_AGV", quando

ela atravessar o bloco *Halt* fará aquela unidade tornar-se inativa e a mesma só voltará a atuar no sistema quando alguma unidade passar por um bloco *Activate*.

#### 4.3.15. O Bloco Move

Este bloco avança um transportador de uma estação para outra sem mover a entidade que o controla para o destino. Esta permanece no bloco atual até que o transportador tenha atingido o destino. A entidade está então habilitada a se mover para outro bloco ou tarefa no sistema.

O tempo gasto para a movimentação do transportador de uma estação para a seguinte é baseado na velocidade do mesmo, especificada no bloco *Transporter*, e na distância entre as estações, definida no bloco *Distance*.

A entidade deve ter obtido controle sobre o transportador, seja através de um bloco *Request*, seja através de um bloco *Allocate*, antes que possa movimentá-lo.

Se o número da unidade do transportador a ser movimentado não for especificado, será utilizada aquela que foi mais recentemente alocada ou requisitada (*Allocated or Requested*).

A entidade permanecerá no bloco *Move* até que o transportador atinja a estação destino.

Neste momento o transportador "AGV" é movimentado de qualquer estação em que esteja para a estação "Base".

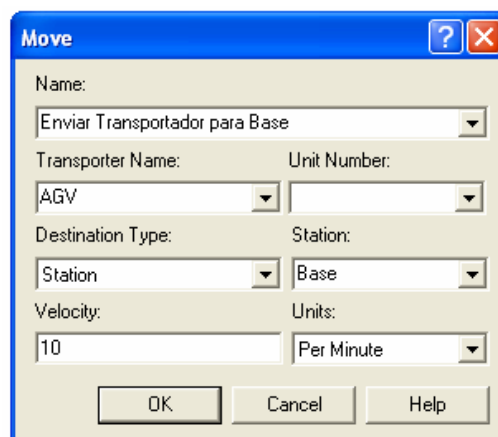


Figura 56 - Exemplo de configuração do bloco Move

#### 4.3.16. O Bloco Request

O bloco *Request* designa uma unidade de um transportador a uma entidade e o move para a estação onde ela esteja. Pode ser especificada uma entidade específica do transportador ou a seleção pode ser baseada em uma regra.

Quando uma entidade chega ao bloco, um transportador é alocado quando estiver disponível. A entidade permanece no bloco *Request* até que a unidade do transportador tenha atingido sua estação.

A entidade que entra no bloco *Request* deve ter um valor válido no seu atributo *Entity.Station*. Isso para evitar o mesmo problema que já foi explicado no bloco *Access* que é causado por especificações feitas nos blocos *Batch* e *Separate* principalmente.

O bloco *Request* é uma combinação dos blocos *Allocate* e *Move* no sentido de reservar o transportador para uma entidade e ao mesmo tempo já movimentá-lo para onde quer que ela esteja.

Figura 57 - Exemplo de configuração do bloco Request

A entidade ao passar por esse bloco então vai selecionar uma das unidades disponíveis do transportador "AGV" e guardar o seu código no atributo "Numero\_AGV". Onde quer que essa unidade esteja, ela será movimentada até a estação onde estiver a entidade que a requisitou.

A velocidade com que o transportador se movimentará será de 10 unidades de distância por minuto e esse será seu valor dali pra frente até que seja "trocado" por alguma outra especificação em algum bloco do tipo *Move* ou *Transport*.

A prioridade dessa entidade para requisitar o transportador é a padrão, uma vez que não foi especificada e igual a "1". Outras entidades em blocos *Request* (ou *Allocate*) que estiverem tentando tomar controle do transportador e tenham prioridades menores do que "1" serão atendidas antes que a que estiver no bloco do exemplo.

#### 4.3.17. O Bloco Transport

Este bloco transporta tanto a entidade controladora quanto o transportador em questão entre duas estações.

O tempo necessário para a movimentação de ambos é baseado na velocidade do transportador e na distância entre as estações.

Quando uma entidade entra no bloco *Transport*, seu atributo *Entity.Station* recebe o código ou valor representativo da estação de destino. A entidade é em seguida transportada até essa próxima estação.

Figura 58 - Exemplo de configuração do bloco Transporter

No exemplo da figura 49, a entidade será levada da estação em que está até a estação "Torneamento" pelo transportador "AGV". A entidade permanecerá no bloco *Transport* até que o AGV atinja a estação destino. Em termos de Animação, não será uma figura da entidade que aparecerá seguindo de uma estação a outra, mas sim a do transportador, ao contrário do que acontece na animação de condutores.

#### 4.3.18. O bloco Sequence

O bloco *Sequence* é usado para definir uma seqüência para o fluxo que a entidade seguirá dentro do modelo. Uma seqüência consiste na lista ordenada das estações que a entidade visitará. Para cada estação na seqüência, atributos, variáveis podem ter seus valores atribuídos dentro do bloco seqüência.

Existem três atributos especiais que são atribuídos a todas as entidades. O atributo de seqüência (*Entity.Sequence*) define a seqüência que a entidade deve seguir; um valor igual a 0 significa que a entidade não está seguindo seqüência nenhuma. Para que uma entidade siga uma seqüência, seu atributo seqüência deve ser atribuído um valor (por exemplo, através de um bloco *Assign*). O atributo *Jobstep* (*Entity.Jobstep*) guarda a posição da entidade dentro da seqüência a ser seguida. Esse valor é automaticamente atualizado cada vez que uma entidade é transferida devido a seqüência. O atributo *PlannedStation* (*Entity.PlannedStation*) guarda o número da estação associada com o próximo *jobstep* na seqüência.

#### 4.3.19. O bloco Conveyor

O bloco *Conveyor* permite a definição de um condutor acumulativo ou não acumulativo para uma entidade em movimento entre *Stations*

A velocidade, tamanho da célula, e comprimento dos segmentos, devem usar uma unidade de comprimento comum.

Condutores não acumulativos, ao sofrerem uma parada para carregar ou descarregar uma entidade, faz com que todas as entidades parem de forma que o espaçamento entre as entidades no condutor é constante.

Ao usar um condutor acumulativo, este não sofre paradas ao carregar ou descarregar uma entidade. Um bloqueio local ocorre e entidades continuam sendo conduzidas até o bloqueio. Neste momento o tamanho da acumulação é avaliado.

A soma das distâncias entre as estações ligadas pelo condutor deve ser divisível pelo tamanho da célula. Caso contrário acontecerá um erro de verificação do modelo.

Entidades serão acumuladas baseado no "*Accumulation Size*" especificado. Este tamanho pode ser diferente do número de células que a entidade originalmente acessa. Neste caso a entidade retornará ao espaço original quando o bloqueio desaparecer.

A velocidade do condutor pode ser mudada usando a variável *VC(Conveyor)*.

Existem muitas variáveis de estado para condutores tanto acumulativas como não acumulativas. Algumas são comuns a ambos os tipos como, por exemplo, *LEC(Conveyor)* e *NEC(Conveyor)* que avaliam o comprimento e o número de entidades sendo conduzidas pelo condutor respectivamente.

#### 4.3.20. O bloco Segment

O bloco *Segment* permite a definição de segmentos que representam o caminho percorrido pelo condutor. O caminho de um condutor é composto por uma série de segmentos conectados onde cada segmento é um link direto entre duas *Stations* e o caminho é definido por uma estação de início e um grupo de distâncias até as próximas estações.



O comprimento dos segmentos deve ser definido em unidades inteiras de acordo com as unidades. A unidade de medida não é importante desde que sejam usadas unidades consistentes entre os blocos *Segment* e *Conveyor*.

É importante notar que os segmentos são direcionais de forma que as entidades só se movem entre a estação de início e a próxima estação.

#### 4.3.21. O bloco Transporter

O bloco *Transporter* permite a definição de transportadores livres ou guiados para uma entidade se movendo de um local para o outro dentro do modelo. Transportadores livres se movem de um local para o outro sem ser influenciados pelo tráfego de outros transportadores. O bloco *Distance* especifica a distância entre estações. Transportadores guiados estão restritos a caminhos fixos como, por exemplo, trilhos. O movimento dos mesmos pode ser afetado pelo tráfego ou congestionamento de veículos. Transportadores guiados utilizam redes definidas usando os blocos *Network* ou *Network Links*.

Todas as unidades de distância especificadas no modelo devem ser consistentes. A velocidade do transportador pode ser alterada dentro do modelo atribuindo a mesma um novo valor através da variável  $VT(Transporter)$  para cada grupo de transportadores ou através da variável  $VTU(Transporter, Unit Number)$  para uma unidade específica de transporte. Adicionalmente a velocidade pode ser temporariamente mudada nos módulos *Move*, *Request*, e *Transport*.

Variáveis específicas dos transportadores como  $NT(Transporter)$  e  $MT(Transporter)$ , podem ser usadas para determinar o número de transportadores ocupados ou ativos, respectivamente. Variáveis adicionais podem ser usadas para verificar o status de unidades de transporte individuais.

#### 4.3.22. O bloco Distance

O bloco *Distance* define uma distância entre duas estações que fazem parte de um conjunto de distâncias para um transportador livre. A estação de início, estação de final e distância serão utilizadas para criar um grupo de distâncias apropriado, que será usado durante a simulação pelo transportador que se move entre as duas estações.

As distâncias devem ter valores inteiros não negativos. Se as distâncias reais não são quantidades inteiras da unidade de medida selecionada e utilizada por todos os parâmetros dos transportadores (velocidade e distância), devem ser escalonadas de forma a ser distâncias inteiras ou as distâncias devem ser arredondadas para o inteiro mais próximo.

Se um par de estações (Estação de início – Estação de final) aparece apenas uma vez no "set" de distâncias, é assumido que a distância entre estação final e estação de início é igual a distância estação de início – estação de final. Caso contrário, serão utilizados os valores especificados para cada um dos casos.

Se o transportador caminha entre duas estações cuja distância não se encontram especificada no bloco *Distance*, uma distância de zero unidades é assumida pelo transportador e este não será mostrado na animação.

#### 4.3.23. O bloco Network

O bloco *Network* define um mapa que um grupo de transportadores guiados deverá seguir. Uma rede engloba o grupo de links especificados no bloco *Network Links* associado. Os parâmetros dos links serão todos especificados no bloco *Network Link*.

#### 4.3.24. O bloco Network Link

O bloco *Network Link* define as características de das opções de caminho determinadas para um transportador guiado. O bloco *Network* referencia esse bloco de forma a informar ao transportador as opções de caminho disponíveis.

#### 4.3.25. O bloco Activity Área

O bloco *Activity Área* permite definir uma área que pode ser associada com uma ou mais estações. Cada área de atividade pode ter uma área de “atividade pai” associada, que será usada para definir a hierarquia do sistema modelado. Estatísticas são automaticamente geradas de acordo com a hierarquia das área de atividades.

Uma área de atividade coleta todas as estatísticas de custo e tempo associadas com estações componentes da mesma ou “áreas de atividade filhas”.

#### 4.3.26. Diferenças entre blocos

A fim de se compactar as informações descritas neste item 4, é apresentado abaixo um resumo para consulta rápida e facilitada:

##### *4.3.26.1. O Bloco Transport*

Este bloco transporta tanto a entidade controladora quanto o transportador em questão entre duas estações.

##### *4.3.26.2. O Bloco Move*

Este bloco avança um transportador de uma estação para outra sem mover a entidade que o controla para o destino. Esta permanece no bloco atual até que o transportador tenha atingido o destino. A entidade está então habilitada a se mover para outro bloco ou tarefa no sistema.

##### *4.3.26.3. O Bloco Route*

Este bloco transfere uma entidade para uma estação específica, ou para a próxima estação em uma visitação seqüencial de estações definida para a entidade. Um atraso para a transferência pode ser definido.

##### *4.3.26.4. O Bloco Allocate*

Este bloco designa um transportador a uma entidade sem que essa necessariamente desloque-se junto com ele e sem que o transportador mova-se até onde está a entidade. Ela então pode conduzi-lo até um ponto específico ou pará-lo.

#### *4.3.26.5. O Bloco Request*

O bloco Request designa uma unidade de um transportador a uma entidade e o move para a estação onde ela esteja. Pode ser especificada uma entidade específica do transportador ou a seleção pode ser baseada em uma regra.

#### *4.3.26.6. O Bloco Free*

Este bloco libera o transportador mais recentemente utilizado pela entidade. Se não existem entidades aguardando a liberação do transportador, este ficará ocioso na estação da entidade onde ocorreu a liberação, a menos que esteja especificado diferentemente no bloco "Transporter".

## 5. Outros recursos de Modelagem

### 5.1. Submodelos

Usar submodelos facilita a visualização do sistema como um todo, permite esconder elementos repetitivos, e obter estatísticas de blocos de lógica como um todo e não individualmente, além de ser simplesmente uma maneira muito eficaz de organizar o seu processo.

Os submodelos podem ser conectados a outros módulos, outros submodelos, ou simplesmente bastar-se por si mesmo. Simplesmente edite as propriedades (Properties) do submodelo para alternar entre essas características. Você pode criar um “objeto” submodelo vazio, abri-lo para mostrar a sua “janela” de trabalho e colocar seu fluxo lógico, gráficos ou animações dentro dele. Ou você pode também pegar a sua lógica que já está pronta e “jogar” para dentro de algum submodelo através do comando *Aggregate*. Ou, pelo processo inverso, retirar a lógica de um submodelo e mandá-lo para níveis superiores no modelo geral através do comando *Unaggregate*.

Um pequeno ponto a ser observado é que, se um módulo *Batch* é utilizado para fazer um grupamento do tipo temporário, então a entidade representativa do conjunto deve ser posteriormente desagrupada utilizando-se um bloco *Separate* antes que deixe o submodelo para voltar para o nível hierárquico superior ou mesmo descer para um próximo submodelo.

### 5.2. Diferentes “Vistas” para o Modelo

Existe um comando no Arena que se chama Named Views (algo como “vistas nomeadas”) que permite que você utilize alguma tecla de atalho para referenciar alguma “vista” do seu modelo, alguma parte específica do mesmo.

Uma vez que você dê o comando de nomear uma certa parte do seu modelo, o que estiver sendo mostrado naquele exato momento na janela de trabalho será considerado como a “visão” nomeada. O mesmo posicionamento, a mesma quantidade de zoom e todas as outras características “visuais”, serão mantidas e referenciadas por aquele nome ou por aquela tecla de atalho que foi escolhida.

Para fazê-lo, clique em: **View → Named Views → Add**, e então escolha a tecla de atalho e o nome da “Vista”, e uma vez que elas tiverem sido adicionadas ao modelo, aparecerão no Painel de Navegação (Navigate Panel), então basta clicar sobre a vista em particular desejada, ou simplesmente apertar a tecla de atalho.

### 5.3. O Painel de Navegação

O painel de navegação presente na Barra de Projeto (Verticalmente à esquerda na tela principal do software) proporciona uma maneira simples de navegar através das diferentes Named Views e dos diferentes níveis de hierarquia (submodelos), de maneira similar ao Windows Explorer.

Quando uma Named View é adicionada ao modelo seu nome e tecla de atalho são adicionados ao painel de navegação e assim também acontece quando novos submodelos são criados, seu nome também aparece agregado a uma seta que aponta para baixo indicando que é um nível hierárquico inferior.

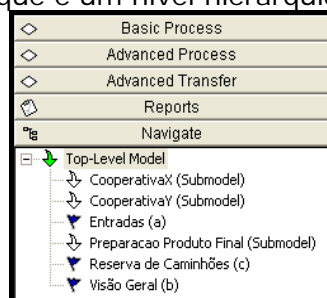


Figura 59 - Painel de Navegação

## 6. Rodando a Simulação e Vendo os Resultados

### 6.1. Preparando uma Seção de Simulação

Antes de iniciar uma simulação, você deve especificar a velocidade com que será executada a animação de entidades, recursos e outros elementos do modelo, selecionar algumas opções de parâmetros da simulação ou configurar o modo da apresentação de relatórios ao final da execução. Além disso, dados como quais estatísticas deverão ser coletadas, e os parâmetros de cada replicação, como sua duração ou a unidade-base de tempo também devem ser especificados. Isso é feito em uma janela acessada através do caminho: **Run->Setup...**

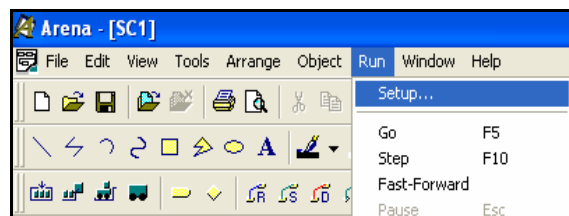


Figura 60 - Acesso a janela de Setup de simulação

### 6.2. Iniciando e Terminando Seções de Simulação

Uma vez que seu modelo já esteja pronto e você queira colocá-lo para “funcionar”, vá até o submenu “Run” no menu principal, no topo da tela e escolha o comando “Go”, ou simplesmente aperte o botão que se assemelha à tecla “Play” da grande maioria dos equipamentos eletrônicos (conforme figura abaixo). Esse botão está localizado na barra de ferramentas padrão.



Figura 61 - Botões para controle da simulação presentes na barra de ferramentas

Isso feito, o Arena checará o seu modelo para garantir que ele seja válido, que não haja blocos que não tenham sido conectados, ou não tenham sido editados, ou que contenham duplicidade de nomenclatura, entre outros erros que podem acontecer.

Quando você começa a executar uma simulação, o software entra em um estado especial chamado “Run Session”, no qual ele permanece até que seja explicitamente interrompido através do comando “End” no submenu “Run” ou ao se apertar o botão que se assemelha à tecla “Stop”. Uma vez em “Run Session”, vários comandos, módulos e barras de ferramentas ficarão indisponíveis, só podendo ser acessados após o fim da Seção. Vale ressaltar que uma vez que um arquivo do Arena esteja em “Run Session”, outro arquivo não poderá iniciar sua simulação simultaneamente.

Quando uma seção de simulação chega ao final, se você tiver configurado o modelo para automaticamente mostrar os relatórios, ele o fará nesse momento. Em qualquer outro caso, você poderá escolher mostrar um ou mais tipos de relatórios listados no painel de relatórios à esquerda da tela.

### 6.3. Setup

Essa janela é atingida através do seguinte caminho (o mesmo mostrado na figura da página anterior) e é utilizada para mudar várias das opções de configuração antes de iniciar uma seção de simulação. Uma janela aparecerá contendo cinco categorias de opções: Parâmetros de Projeto, Parâmetros de Replicação, Velocidade de Simulação, Controle da Simulação e Relatórios.

#### 6.3.1. Parâmetros de Projeto

Essa categoria enquadra as informações básicas do projeto de simulação e inclui dados como qual o título do projeto e o nome do analista que serão mostrados nos relatórios, assim como incluem quais os tipos de dados estatísticos que deverão ser coletados durante a execução.

É interessante especificar o título do Projeto para facilmente identificar os relatórios de diferentes “variações” que estão sendo avaliadas em cima de um mesmo modelo.

Existem “Checkboxes” para selecionarem-se os tipos de estatísticas que serão coletadas, sejam elas custos, entidades, recursos, filas, processos, transportadores e condutores. Os detalhes de cada uma dessas categorias são descritos abaixo:

Custos (*Costing*), se selecionada, fará aparecer no relatório final todos os dados relativos a atributos de tempo e custos de cada entidade como, por exemplo, *Entity.Vatime*, *Entity.VACost* (onde VA é abreviação de *Value-Added* ou Valor Agregado), *Entity.WaitTime*, *Entity.WaitCost* e outros, permitindo então o cálculo dos custos relativos a cada processamento, ou a cada entidade “produzida”.

Entidades (*Entities*), se selecionada, vai gerar um relatório com todos os tipos de tempos relativos a cada entidade em específico, como tempo em que há geração de valor agregado, tempo em que não há geração de valor agregado, tempo de espera, tempo em transferências e outros tipos de tempo, de forma que intervalos e médias possam ser calculadas e mostradas nos relatórios.

Recursos (*Resources*), se selecionada, vai permitir obter dados como número de recursos ocupados ou a taxa de utilização de cada recurso específico.

Filas (*Queues*), se selecionada, permitirá obter dados como o tempo de espera em cada fila, o número médio de pessoas naquela fila, entre outros.

Processos (*Processes*) referem-se a estatísticas relativas aos processos definidos dentro de cada módulo com o mesmo nome, como o tempo médio de processamento, número de entidades processadas, entre outros.

Transportadores (*Transporters*) funcionam muito parecido com as estatísticas dos recursos, ou seja, número de transportadores utilizados e taxa de utilização de cada unidade específica, entre outras.

Condutores (*Conveyors*) permitem obter dados como taxa de utilização, tamanho da “fila” acumulada quando há bloqueios, tempo de bloqueamento, entre outros.

#### 6.3.2. Parâmetros de Replicação

Esses parâmetros são especificado com informações sobre cada replicação dentro de um projeto de simulação. Inclui-se então o número de replicações a serem executadas, a duração ou “tamanho” de cada uma delas, a existência ou não de um período de “aquecimento”, a unidade-base de tempo, e o tipo de inicialização a ser utilizada entre cada replicação.

**Número de Replicações** – Deve conter um valor inteiro e maior do que “1”. O número ideal a ser utilizado deve ser calculado através de fórmulas aprendidas na parte teórica da disciplina de “Avaliação de Desempenho”.

**Período de Aquecimento** (*Warm-up*) – É um tempo a mais que é incluído no início da replicação, em que depois de seu término, as estatísticas são “zeradas” e o sistema passa a funcionar a partir dali. É utilizado para que o sistema não comece a contar do “zero”, pois alguns tipos de estatísticas ficariam comprometidos, como por exemplo o taxa de utilização. Se o tempo médio começa a ser contado desde o início, um recurso que só é alcançado quando as entidades alcancem o fim do modelo terá a sua taxa de utilização um pouco menor que no sistema real porque ficará “parado” até que as entidades atravessem todo o fluxo. Caso o sistema tenha um período de aquecimento, a hora que as estatísticas começarem a ser coletadas, já existirá alguma entidade sendo processada por esse recurso.

**Unidade de Tempo** – Usado para definir a unidade de tempo (minutos, segundos, horas, etc) para o período de aquecimento.

**Tamanho da Replicação** – Quantidade de tempo em que a simulação ficará rodando. Atenção para o fato: tempo “simulado” ou “computacional” e não tempo “real”. Se nenhum valor for especificado, a opção padrão é que ele rode infinitamente, por esse motivo é sempre bom verificar esse parâmetro antes de iniciar a simulação. Há outras maneiras de se parar uma replicação, quais sejam elas, especificar um número máximo de entidades “criadas” no sistema, especificar uma condição de término ou definir um limite em algum contador a ser especificado em um módulo Statistic ou em um elemento Counters.

**Unidade de Tempo** – Usado para definir a unidade de tempo (minutos, segundos, horas, etc) para a replicação como um todo.

**Horas por dia** – Define um número de horas a ser considerado dentro de um “dia simulado”. Este campo é útil para “excluir” da simulação a parte do dia em que o sistema modelado não seja utilizado, como por exemplo, uma fábrica que trabalha em dois turnos de quatro horas possui um dia de 8 horas apenas e não 24 horas. Simulando-se uma semana desse sistema (5 dias úteis) teríamos um “Tamanho de Replicação” de 40 horas.

**Unidade-Base de Tempo** – É a unidade de tempo que será considerada para os relatórios, para a barra de status, para avaliação da variável TNOW e para gráficos animados que sejam em função do tempo. Todos os atrasos, tempo de replicação, tempo de aquecimento e outros serão convertidos para essa unidade pelo software.

**Condição de Término** – Especifica-se uma expressão que se “Verdadeira” ou avaliada em “1”, termina a simulação. Esse é um método alternativo ao “Tamanho da Replicação” para fechamento de seção de simulação.

**Estatísticas (Statistics)** – Determina se os dados estatísticos obtidos em uma replicação vão ou não ser mantidas entre uma replicação e outra.

**Sistema (System)** – Se selecionada essa opção, após cada replicação, todas as entidades são removidas do sistema, e o status de cada elemento e as variáveis são devolvidas aos seus valores originais.

### 6.3.3.Velocidade de Simulação

No Arena, a animação e a velocidade de animação de um modelo são determinados por diversos fatores, incluindo a frequência com que os “frames” de animação são atualizados e a frequência com que interrupções do teclado são checadadas.

A animação no Arena é muito similar a de um filme, no qual são criados vários “quadros, em que cada um deles representa uma vista levemente alterada do sistema. A velocidade de animação depende então do tamanho do intervalo de tempo simulado que existe entre a “captação” de um quadro e outro e a taxa (em tempo real) com que os quadros são exibidos.

### 6.3.4.Relatórios

Os relatórios podem ser gerados ao fim da execução da simulação, clicando-se em qualquer um dos sub-relatórios listados no painel apropriado (Reports Panel), ou fazendo-se o mesmo quando o modelo está apenas pausado durante uma execução. Além disso, o Arena pode ser configurado para automaticamente mostrar um relatório específico quando a execução chega ao fim.

A janela Reports acessada através do comando Setup no submenu Run permite estabelecer o que acontecerá, em termos de relatórios, o que acontecerá ao final da simulação.

O campo relatório-padrão é onde você identifica qual será aquele que vai ser usado tanto para o caso de sempre mostrar ou para o caso de perguntar antes de mostrar. A caixa de escolha mostra quais são os disponíveis – são os mesmos que estiverem listados no Reports Panel. Além disso, há um relatório que será sempre listado no final chamado de SIMAN Summary (arquivo \*.out). Esta opção permite a você mostrar uma saída em formato texto que é sempre gerada de qualquer maneira pra cada arquivo \*.doe do Arena. O arquivo \*.out é um formato diferente dos relatórios normais e não é mostrado na janela de relatórios como os outros. Se a opção de usá-lo é selecionada, você será chamado a identificar um programa que possa mostrá-lo, sendo que qualquer editor de texto pode fazê-lo e o programa padrão escolhido é o “Bloco de Notas”.



## 7. Identificando e Corrigindo Erros

Quando você inicia uma seção de execução de simulação (através do comando Go, por exemplo), o Arena automaticamente checa os dados de entradas e as animações do seu modelo. Este passo pode detectar erros na sua modelagem. Um erro (error) refere-se a algum problema com a lógica ou os dados, enquanto que um aviso (warning) refere-se a um problema com a animação.

Se não há a ocorrência de nenhum erro durante o processo de checagem, o Arena continuará com a execução. Quaisquer erros que ocorram daí para frente serão considerados erros de execução (runtime errors).

Se o Arena detectar um erro no seu modelo, ele vai avisá-lo através da janela Errors/Warnings. Esta caixa tem uma área de mensagens para descrever melhor os erros ou avisos e alguns botões para que você atue na resolução dos mesmos.

### 7.1. Janela Errors/Warnings

Na maioria dos casos, o ARENA pode determinar qual módulo ou elemento de animação está causando o erro no seu modelo. Se isso acontecer, os botões Find e Edit estarão ativos e você poderá usá-los para encontrar o problema e solucioná-lo imediatamente. Essa é uma coisa muito comum de acontecer durante a modelagem, preste atenção nas mensagens de erro e facilmente tudo será resolvido.

### 7.2. Barra de Debug

A barra de Debug é uma inovação da versão 9.0 o ARENA e permite visualizar variáveis, estabelecer breakpoints para simulação, visualizar o código SIMAN gerado a cada etapa da simulação, etc. Para visualizar a barra de Debug na Interface, basta selecionar a opção de Menu **View->Debug Bar**. Ao fazer isso aparecerá no canto inferior da tela o campo mostrado na figura 63.

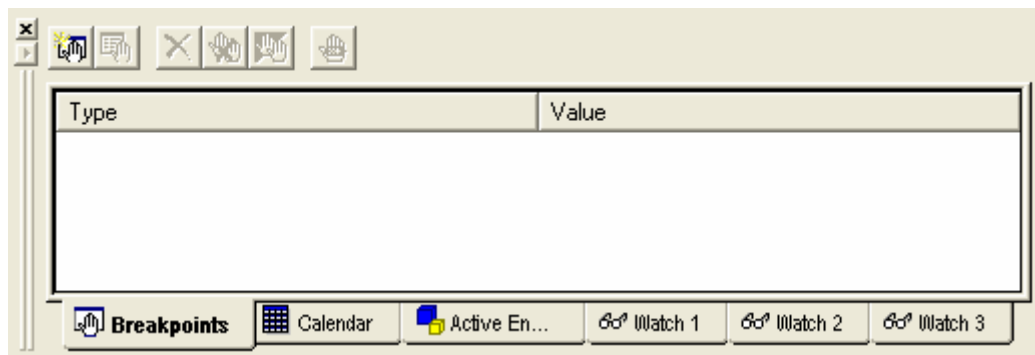


Figura 62 - Barra de Debug

#### 7.2.1. Breakpoints

Permite controlar a execução da simulação. Breakpoints podem ser adicionados ao arquivo do modelo e mostrados em uma tabela. As paradas geradas pelos Breakpoints podem ser especificadas em termos de tempo, em número de entidades ativas, em módulos ou através de expressões condicionais. Com essas opções é possível examinar operações e lógica do modelo, avaliar mudanças no status do sistema ou fazer apresentações para terceiros.

Os seguintes botões (botões presentes no topo da janela de) permitem controlar os breakpoints do sistema.

**Novo Breakpoint:** permite acessar a janela de diálogo para adição de novos breakpoints ao sistema.

**Breakpoints Properties:** permite editar as propriedades do Breakpoint selecionado.

**Remove Breakpoint:** permite apagar o breakpoint selecionado.

**Disable All:** desabilita todos os breakpoints do modelo. Os breakpoints desabilitados ainda fazem parte do modelo mas são ignorados durante a simulação. Os breakpoints também podem ser desabilitados através de clique com o mouse nos checkboxes específicos presentes na tabela de breakpoints.

**Module Break:** adiciona ou remove breakpoints correspondentes aos módulos selecionados na janela.

Algumas observações ainda podem ser feitas a respeito dos breakpoints. Primeiramente, quando a execução do modelo é interrompida, a janela de Breakpoints é automaticamente apresentada e a linha correspondente ao breakpoint que gerou a interrupção é destacada. Além disso, para apagar um breakpoint é possível apenas selecioná-lo e pressionar a tecla delete.

### 7.2.2. Calendar Window

A janela de calendário permite visualizar eventos futuros agendados pelo calendário de eventos SIMAN do ARENA durante a execução de uma simulação. O tempo dos eventos, número de entidades associadas com os eventos e a descrição dos mesmos são apresentados em forma de tabela.

A janela de eventos é atualizada sempre que a simulação é interrompida.

### 7.2.3. Activity Entities Window

A janela de atividade de entidades apresenta o número e valor dos atributos de uma entidade ativa (caso haja alguma) na organização em forma de árvore.

Da mesma forma que a janela de calendário, a janela de atividades de entidades será atualizada sempre que a simulação sofrer uma pausa e permite a alteração de valores de atributos durante uma pausa.

### 7.2.4. Watch Windows

Estão disponíveis três janelas de Watch idênticas na Barra de Debug que permitem monitorar o valor de qualquer expressão na simulação.

Para adicionar expressões a uma janela watch, basta clicar no campo expressão do item desejado na lista de expressões. É possível utilizar o Expression Builder para buscar variáveis ou construir expressões que devam ser observadas.

Além de expressões, objetos da *Runtime Elements bar* (*resource, transporter, variable, queue, etc*) podem ser adicionados a janelas Watch. Para adicionar o objeto basta copiar e colá-lo dentro da janela ou arrastá-lo da *Runtime Elements Bar* para a janela de Watch.

A expressão a ser observada é atualizada sempre que a simulação sofre uma pausa.

## 8. Interpretando Resultados/Relatórios

O Painel de Relatórios (Reports Panel) na barra de Projeto, localizada verticalmente à esquerda da tela principal do Arena, lista os vários relatórios disponíveis com os resultados das simulações. Clicando em qualquer um deles faz aparecer uma nova janela com o relatório selecionado para o modelo que tiver sido executado.

Há oito tipos de relatórios fornecidos pelo Arena. São eles: *Category Overview*, *Entities*, *Processes*, *Queues*, *Resources*, *Transfers*, *User Specified*, *Frequencies*. Em cada um destes são apresentados cálculos estatísticos para as variáveis do modelo que foram selecionadas para constar nas estatísticas.

Os cálculos são feitos levando em conta o número de replicações.

### 8.1. Tipos de cálculos de estatísticas

Quando é executada apenas uma replicação ou quando esteja observando-se uma replicação específica:

**Média** (*Average*) – a média dos valores obtidos para algum dado em toda uma replicação;

**Ponto Médio** (*Half Width*) – calculado entre os valores obtidos para algum dado em toda uma replicação;

**Valor Mínimo** (*Minimum Value*) – o menor valor observado para alguma variável ou dado específico;

**Valor Máximo** (*Maximum Value*) – o maior valor observado para alguma variável ou dado específico;

Quando ocorrem várias replicações:

**Média** (*Average*) – a média de todas as médias de cada replicação em cima de determinado dado;

**Ponto médio** (*Half Width*) – calculado como uma média entre o ponto médio de cada replicação;

**Valor Médio Mínimo** (*Minimum Average*) – a menor média sobre algum dado entre as médias de todas as replicações;

**Valor Médio Máximo** (*Maximum Average*) – a maior média sobre algum dado entre as médias de todas as replicações;

**Valor mínimo** (*Minimum Value*) – o menor valor identificado de algum dado em todas as replicações;

**Valor Máximo** (*Maximum Value*) – o maior valor identificado de algum dado em todas as replicações.

### 8.2. Contadores e Saídas

No que diz respeito a contadores e saídas, os resultados apresentados são calculados conforme veremos a seguir.

Quando é executada apenas uma replicação ou quando esteja observando-se uma replicação específica:

**Valor** (*Value*) – o valor final ou mais atualizado (caso a simulação tenha sido interrompida antes da conclusão) do contador ou da saída.

Quando ocorrem várias replicações:

**Média** (*Average*) – a média dos valores finais ou de contadores entre todas as replicações;

**Ponto médio** (*Half Width*) – calculado como o ponto médio entre os valores de cada replicação;

**Valor mínimo** (*Minimum Value*) – o menor valor identificado de algum dado em todas as replicações;

**Valor Máximo** (*Maximum Value*) – o maior valor identificado de algum dado em todas as replicações.

### **8.3. Variáveis referentes a Entidades**

**Tempo de valor não agregado** (*NVA Time*) – Corresponde ao tempo em que cada entidade do sistema passou em atividades que não agregavam valor (pré-definidas na modelagem).

**Outros tempos** (*Other Times*) – Corresponde ao tempo em que cada entidade do sistema passou em atividades que não estivessem encaixadas como de valor agregado, sem valor agregado, de espera ou transferência.

**Tempo total** (*Total Time*) – Corresponde ao tempo em que cada entidade passou dentro do sistema.

**Tempo de Transferência** (*Transfer Time*) – Corresponde ao tempo em que cada entidade do sistema passou em um transportador, condutor, rota ou qualquer processo ou atraso que tenha seu tempo alocado no tipo transferência.

**Tempo de valor agregado** (*VA Time*) – Corresponde ao tempo em que cada entidade do sistema passou em atividade de valor agregado.

**Tempo de espera** (*Wait Time*) – Corresponde ao tempo em que cada entidade do sistema passou em filas ou qualquer processo ou atraso que tenha seu tempo alocado no tipo espera.

**Número de entidades que entraram no sistema** (*Number In*) – soma das entidades de cada tipo que foram “criadas” nos blocos Create disponíveis ou através de duplicação em blocos Separate.

**Número de entidades que deixaram o sistema** (*Number Out*) – soma das entidades de cada tipo que deixaram o sistema através de um bloco Dispose.

**Trabalho em Desenvolvimento** (*WIP*) – soma das entidades de cada tipo que permaneceram dentro do sistema em algum processamento ao final da simulação ou replicação.

### **8.4. Variáveis referentes a Filas**

**Tempo de Espera** (*Waiting Time*) – Corresponde ao tempo de espera em cada fila, o qual é definido como o período de tempo desde que a entidade entra na fila até a hora em que ela a deixa. É mostrado o menor valor, maior valor e o valor médio do intervalo de tempo, para cada fila do sistema.

**Quantidade em espera ou tamanho da fila** (*Number Waiting*) – Corresponde ao tamanho máximo que a fila alcançou em determinado período de tempo.

### **8.5. Variáveis referentes a Recursos**

**Quantidade ocupada** (*Number Busy*) – Refere-se à média no tempo do número de unidades de determinado recurso que se mantiveram ocupadas.

**Taxa de Utilização** (*Utilization*): Refere-se a uma ponderação entre o tempo que o recurso passou sendo utilizado e o tempo em que passou ocioso.

**Número de vezes em que foi usado** (*Number Times Used*): Refere-se ao número de vezes em que cada recurso foi reservado (*seized*).

### **8.6. Variáveis referentes a Condutores**

**Bloqueado** (*Blocked*) – refere-se ao tempo em que o condutor manteve-se ativo e com entidades sobre ele, mas que elas não estivessem sendo movimentadas por algum bloqueio causada por qualquer uma das entidades (ela estivesse sendo processada sobre o próprio condutor, por exemplo)

**Tamanho Acumulado** (*Length Accumulated*) - o total do tamanho ocupado pelas entidades quando ocorreram bloqueios, ou, em outras palavras, o tamanho da fila sobre o condutor quando que se formou nas ocasiões em que alguma entidade parou sobre o mesmo.

**Taxa de Utilização** (*Utilization*) – a razão entre o tamanho total (ou número de células totais) do condutor e a quantidade (ou o tamanho) de células ocupadas ao longo do tempo.

### **8.7. Variáveis referentes a Transportadores**

**Quantidade ocupada** (*Number Busy*) – idem ao que acontece com os recursos, mas especificamente para as unidades de um transportador.

**Taxa de Utilização** (*Utilization*) - Refere-se a uma ponderação entre o tempo que o transportador passou movimentando-se e sendo utilizado e o tempo em que passou ocioso.

### **8.8. Variáveis referentes a Processos**

**Tempo Acumulado** (*Accumulated Time*) - soma, para cada categoria (valor agregado, espera, etc), do tempo em que cada entidade passou dentro do processo.

**Quantidade que entra** (*Number In*) - o número de entidades que entraram em determinado processo.

**Quantidade que sai** (*Number Out*) - o número de entidades que saíram de um determinado processo.

## 9.0 Process Analyser

O *Process Analyser* ajuda na avaliação de alternativas apresentadas na execução da simulação de diferentes cenários para o modelo do sistema estudado. Ele é bastante útil para desenvolvedores de modelos de simulação, assim como para tomadores de decisões dentro das empresas (que freqüentemente não estão muito familiarizados com a modelagem, mas sim com a solução que a simulação está oferecendo).

O *Process Analyser* é focado na comparação de cenários depois que o sistema já foi completamente traduzido para um “modelo virtual”. Nesse estágio, tal modelo já está completo, validado e configurado apropriadamente para o uso do *Process Analyser*. O papel dessa ferramenta é então o de permitir a comparação entre os resultados retirados do modelo a partir de diferentes dados de entrada.

### 9.1. Definição de Termos

**Controles** – Entradas consideradas que podem afetar significativamente as respostas no modelo de modo que possam ser monitoradas e controladas a partir da saída do mesmo.

**Respostas** – Saídas do sistema que representam medidas de como o modelo comportou-se durante a execução da simulação.

**Cenário** – Uma coleção de controles e respostas aplicados a um dado modelo de simulação.

Os candidatos a Controles são: Variáveis, Recursos (Valores de Capacidade).

Os candidatos a respostas são: Variáveis, Contadores, Saídas, Intervalos de Tempo, Taxa de Utilização de Recurso, Vários outros tipos de dados estatísticos do sistema.

Todos esses itens que acima que forem encontrados no seu modelo serão apresentados como candidatos (ou como controle, ou como resposta, dependendo da natureza de cada um deles) pelo *Process Analyser* a menos que você especifique que eles sejam excluídos.

### 9.2. Procedimento Geral para Análise de Cenários

1. Definir Controles e Respostas durante o desenvolvimento do modelo no Arena.
2. Gerar o Arquivo de Programa do Modelo (arquivo \*.P)
3. Para obter tal arquivo, carregue o arquivo principal do tipo \*.doe no Arena e então selecione a opção Check Mode no submenu Run. Esse procedimento vai gerar o arquivo desejado.

Obs: Se você já rodou uma vez que seja o modelo em questão, o arquivo \*.P já existirá pois o Arena sempre checa o modelo antes de começar uma seção de simulação.

4. Preparar um conjunto de cenários
5. Executar a simulação de um cenário
6. Analisar os resultados
7. Representar os dados resultantes graficamente

### 9.3. Exemplo do Funcionamento do Process Analyser

Para demonstrar o funcionamento do *Process Analyser*, vamos utilizar um modelo bem simples como o do primeiro exercício do curso, em que uma fábrica deveria decidir se comprava um novo torno ou uma nova fresadora. O objetivo final é que uma maior quantidade de peças, independentemente do tipo, sejam produzidas.

#### 9.3.1. Iniciando a Ferramenta

Simplesmente acesse o submenu Tools e logo em seguida Process Analyser, como na figura abaixo.

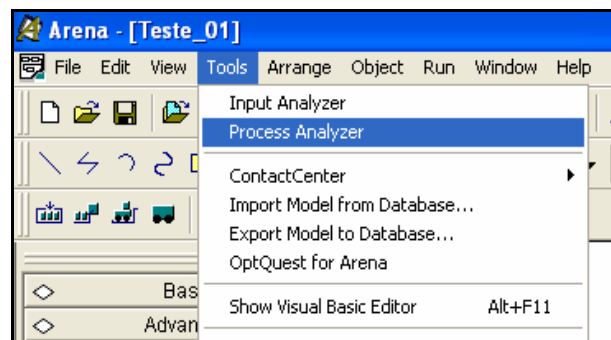


Figura 63 - Acesso ao Process Analyser via Menu Principal do ARENA

#### 9.3.2. Criando um novo projeto

Assim, como no Input Analyser, basta apertar o botão correspondente na barra de ferramentas padrão ou acessar o submenu File e logo depois o comando New, como na figura abaixo.

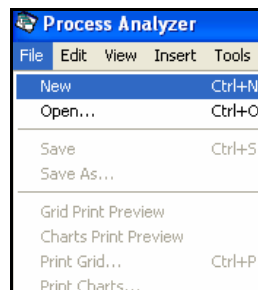


Figura 64 - Criando um novo projeto no Process Analyser

#### 9.3.3. Adicionando um cenário

Para criar um novo cenário basta olhar na tela onde estiver dizendo "Double-click here to add a new scenario" e clicar duas vezes nesse ponto. Bastante didático e intuitivo. Procure pelo arquivo que você está analisando na janela que se abrirá (no caso desse exemplo será o Teste\_01.p). Clique em OK e um novo cenário terá surgido.

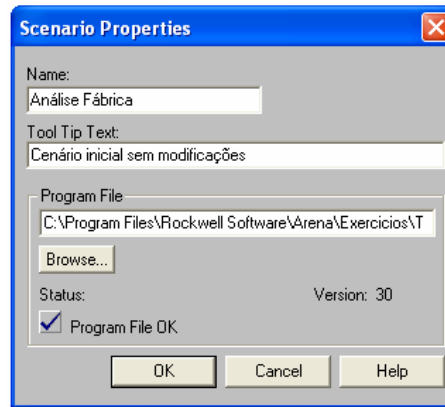


Figura 65 - Definição de cenários

### 9.3.4. Adicionando Controles

Clique com o botão direito sobre o nome do recém-criado cenário e selecione a opção "Insert Control" para inserir um novo controle como na figura abaixo.

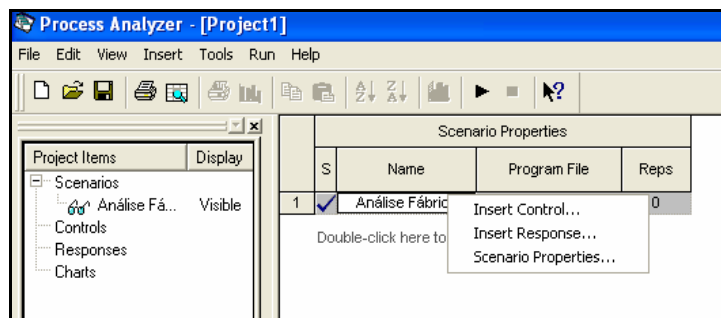


Figura 66 - Acesso ao painel de controles

Na caixa de diálogo dos Controles, clique em Recursos (Resources) ou em Variáveis (Variables) para expandir a árvore de opções e selecionar aquelas disponíveis para serem controles, como na figura abaixo.

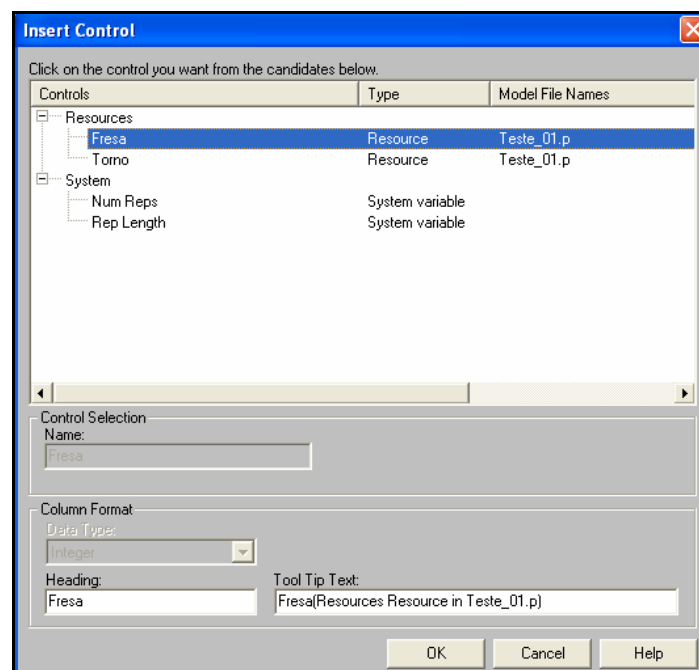


Figura 67 - Configurando controles



Tendo já escolhido os dois controles necessários (a capacidade da fresa e a capacidade do torno), passamos à escolha das repostas.

### 9.3.5. Adicionando Respostas

Mesmo procedimento utilizado para os controles, só que agora você com certeza terá muito mais opções de escolha. De acordo com nosso objetivo inicial, selecionaremos então a resposta “System NumberOut” porque desejamos saber em quanto aumenta o número de peças produzidas no sistema com a mudança nas capacidades do torno e da fresadora.

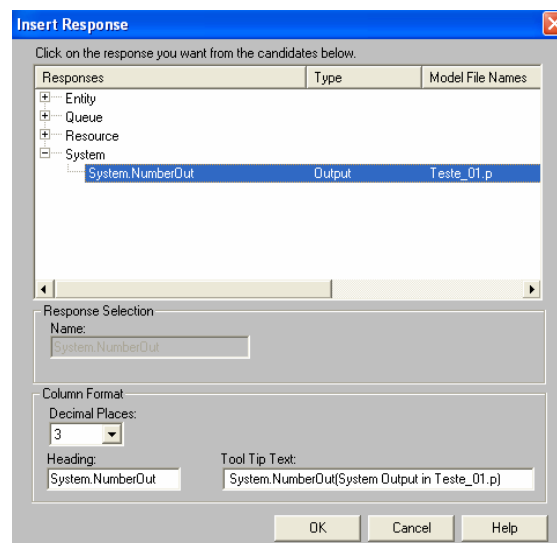


Figura 68 - Configurando Respostas

### 9.3.6. Executando o cenário

Para executar, basta clicar sobre o nome do cenário e escolher o comando Go no submenu Run de maneira análoga a uma execução com um modelo do próprio Arena.

No nosso exemplo, tínhamos definidos anteriormente no próprio modelo que rodaríamos 10 replicações. Ao final dessas, o desenho de uma bandeira aparecerá ao lado do cenário indicando que ele já foi executado. Os valores das respostas já estarão presentes também.

### 9.3.7. Apontando suas “Soluções”

Crie mais cenários sob o mesmo arquivo utilizado no primeiro e apenas altere os valores indicados nos Controles. Para o nosso exemplo, faremos um cenário a mais considerando a presença de 2 tornos e outro cenário considerando a presença de duas fresadoras.

Esses três cenários, já executados, aparecem na figura na próxima página.

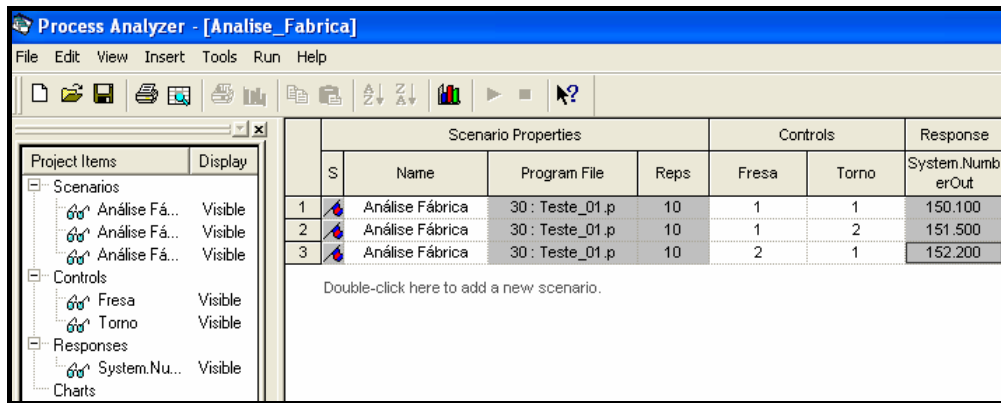


Figura 69 - Process Analyzer após simulação de cenários

### 9.3.8. Transformando seus dados de saída em gráficos

Clique com o botão direito em uma das células com os valores das respostas e selecione a opção "Insert Chart". Na nova caixa de diálogo, ajuste os parâmetros desejados para os gráficos e o resultado deve ser algo como os das figuras abaixo que são aqueles do exemplo dado. E onde se pode inclusive identificar o melhor cenário (em vermelho).

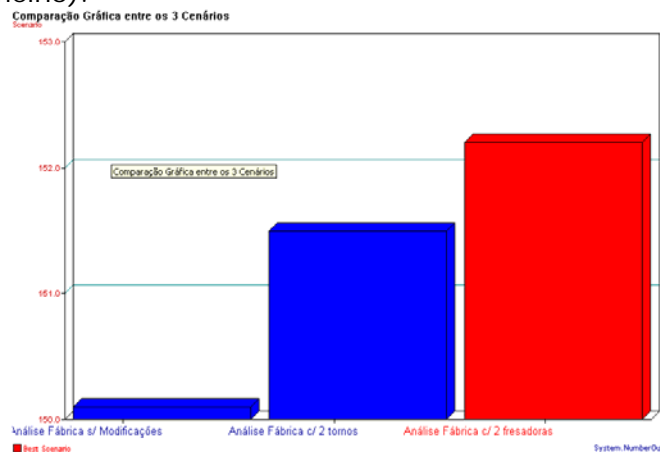



Figura 70 -Gráfico gerado a partir dos resultados de simulação dos vários cenários

## 10. Animação com ARENA 9.0

### 10.1. Animação de Fluxo

A animação de fluxo é a forma mais simples de animação possível no ARENA e implica em mostrar para cada entidade uma figura que se move ao longo dos conectores (linhas que ligam os blocos do modelo ARENA). É possível ligar e desligar a animação de fluxo através do botão , presente na barra de tarefas. A opção de animação de fluxo vem ligada como padrão sempre que se cria um novo modelo.

#### 10.1.1. Entidades

A animação de fluxo consiste basicamente em visualizar as entidades percorrendo seu caminho no modelo através dos conectores entre os vários processos. Assim, o único elemento que compõe a animação além do desenho dos blocos e conectores em si é uma imagem associada à entidade.

A imagem associada a uma entidade é uma propriedade da mesma que pode ser configurada através do atributo Initial Picture através do bloco de fluxo Entity. Para tanto, basta selecionar o bloco de fluxo Entity e então na opção Initial Picture selecionar a imagem desejada. Como Initial Picture é um atributo, a imagem da entidade pode ser alterada ao percorrer modelo utilizando um bloco Assign.

#### 10.1.2. Biblioteca

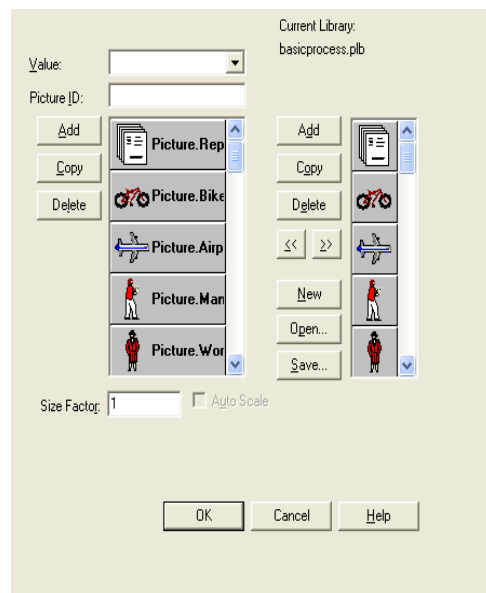
Existem Bibliotecas gráficas com vários tipos de figuras que podem ser utilizadas para substituir a figura padrão utilizada pelo ARENA na animação de fluxo.

Os arquivos das bibliotecas se encontra na pasta ARENA 7.0 dentro do Diretório **C:\Arquivos de programas\Rockwell Software** e tem a extensão \*.plb. As bibliotecas podem ser manipuladas e alteradas através do editor de figuras:

Existem três elementos independentes no editor de figuras. O arquivo da Biblioteca, o gráfico associado à figura de entidade e a figura em si. À esquerda vemos as figuras que possuem um Value e um Picture ID associado. O Value é o que identifica a figura e será utilizado no campo Initial Picture do bloco de fluxo Entity. Picture ID contém uma identificação do gráfico associado à figura. É possível adicionar copiar ou excluir figuras.

A direita pode-se ver os gráficos disponíveis para biblioteca em uso e logo ao lado os botões para manipular os mesmos (add, copy, delete, <<, >>). Os três últimos botões (new, open, save) são utilizados para manipular os arquivos de biblioteca.

Para criar novas figuras, pode-se utilizar os gráficos disponíveis ou criar outros gráficos. Para criar um novo gráfico, pressione o botão add, à direita no painel do editor de figuras. Ao pressionar o botão surgirá um gráfico em branco que pode ser



editado. Para tanto basta selecionar com o mouse o gráfico em branco, abrindo a tela de edição.

Na tela de edição podem ser utilizados os recursos de desenho do ARENA. Ao terminar a edição feche a janela e o gráfico criado estará na Biblioteca. Então, crie uma nova figura, e associe o gráfico criado a ela utilizando o botão <<.

Feita a associação entre o gráfico e a figura, preencha os campos *Picture ID* e *Value*. O campo *Picture ID* deve ser preenchido com uma descrição objetiva do gráfico e o campo *Value* com o nome da figura criada, que será utilizado para associá-la às entidades no modelo.

Obs. Ao criar uma nova figura, o *Value* dela não será mostrado na lista de figuras que aparece no campo *Initial Picture*. Para associá-la a entidade, apenas digite o *Value* no campo *Initial Picture*.

## 10.2. Animação com a barra de ferramentas de animação básica

A barra de ferramenta de animação básica mostrada na figura abaixo, permite animar e visualizar objetos e variáveis dentro de modelos ARENA.



Figura 71 - Barra de ferramentas de animação básica

As ferramentas de animação na ordem em que aparecem são: *clock*, *date*, *variable*, *level*, *histogram*, *plot*, *queue*, *resource* e *global picture placement*.

*Clock*, *date*, *variable*, *level*, *histogram* e *plot* são ferramentas de animação do tipo *status display* e permitem visualizar algum valor associado a variáveis ou aos próprios elementos da simulação como data ou tempo.

### 10.2.1. Clock:

A ferramenta de animação *Clock* permite mostrar o tempo a medida que a simulação progride.

Através do painel de edição é possível determinar:

**Starting time:** a hora que deverá começar a ser contado o tempo correspondendo ao início da simulação em 0.0.

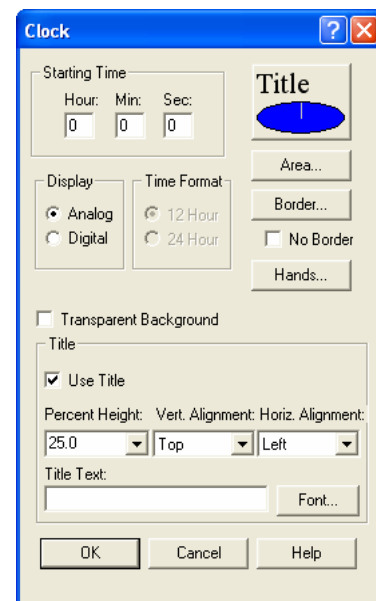
**Display:** tipo de display: analógico ou digital.

**Time format:** formato de tempo: 24 ou 12 horas.

**Title:** caso se opte por utilizar um título (opção *Use Title* marcada), é possível editar o texto, sua formatação, e o posicionamento em relação ao relógio.

É possível ainda configurar a cor através dos botões *Área*, *Border* e *Hands*. As opções *No Border* e *Transparent Background* que dizem respeito às características gráficas do objeto (sem bordas e sem cor de fundo).

As unidades definidas para contar o tempo serão as utilizadas na configuração da simulação em **Run->Setup->Replication Parameters**.



### 10.2.2. Date

A ferramenta de animação *Date* permite mostrar o a progressão do calendário ao longo da simulação.

Através do painel de edição é possível determinar:

**Starting date:** o dia em que deverá começar a ser contada a data correspondendo ao tempo de simulação 0.0.

**Starting time:** a hora que deverá começar a ser contado o tempo correspondendo ao tempo de simulação 0.0.

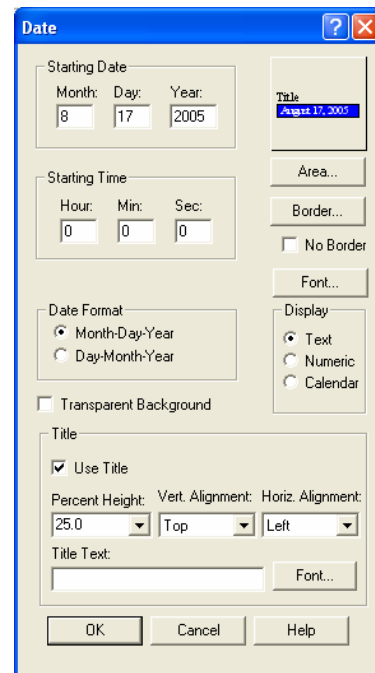
**Display:** formato da informação: texto numérico ou em forma de calendário.

**Date format:** formato da data.

**Title:** caso se opte por utilizar um título (opção *Use Title* marcada), é possível editar o texto, sua formatação e o posicionamento em relação ao relógio.

É possível ainda configurar a cor através dos botões *Área*, *Border* e *Hands*. As opções *No Border* e *Transparent Background* dizem respeito às características gráficas do objeto (sem borda e sem cor de fundo).

As unidades definidas para contar o tempo serão as utilizadas na configuração da simulação em Run > Setup> Replication Parameters (campos Base Time Units e Hours Per Day).



### 10.2.3. Variable

A ferramenta de animação *Variable* permite mostrar o valor (numérico) de uma variável qualquer determinada por uma expressão. A expressão pode ser apenas o nome de uma variável criada dentro do modelo.

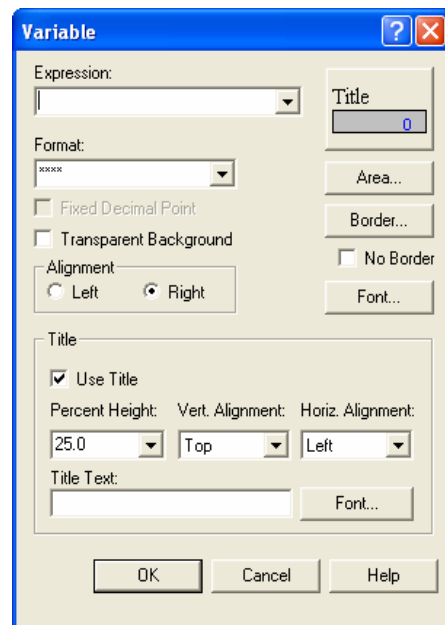
Através do painel de edição é possível determinar:

**Expression:** Determina a variável que será monitorada.

**Format:** Formato do número a ser apresentado.

**Title:** caso se opte por utilizar um título (opção *Use Title* marcada), é possível editar o texto, sua formatação e o posicionamento em relação ao relógio.

É possível, também, configurar a cor através dos botões *Area*, *Border* e *Hands*. As opções *No Border* e *Transparent Background* dizem respeito às características gráficas do objeto (sem borda e sem cor de fundo).



### 10.2.4. Level

A ferramenta de animação *Level* permite mostrar a progressão do valor (numérico) de uma variável qualquer determinada por uma expressão. A expressão pode ser apenas o nome de uma variável criada dentro do modelo.

Através do painel de edição é possível determinar:

**Expression:** Determina a variável que será monitorada.

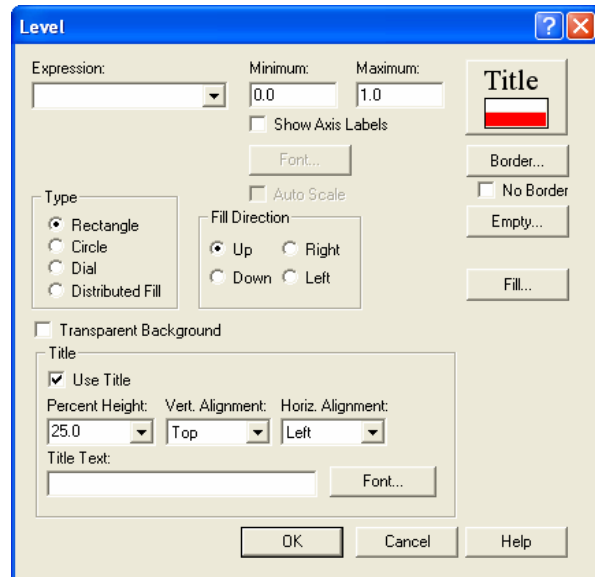
**Minimum e Maximum:** Determina os valores máximo e mínimo do elemento gráfico apresentado na tela.

**Type:** Define o formato do elemento gráfico apresentado na tela.

**Fill Direction:** define a direção em que será preenchido o gráfico.

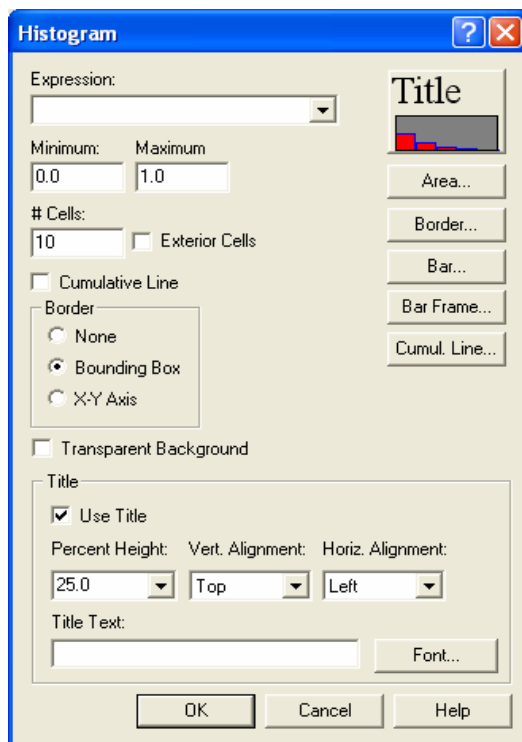
**Title:** caso se opte por utilizar um título (opção *Use Title* marcada), é possível editar o texto, sua formatação, e o posicionamento em relação ao relógio.

É possível, também, configurar a cor através dos botões *Border*, *Empty* e *Fill*. As opções *No Border* e *Transparent Background* dizem respeito às características gráficas do objeto (sem bordar e sem cor de fundo).



### 10.2.5. Histogram

A ferramenta de animação *Histogram* permite mostrar o valor (numérico) de uma variável qualquer determinada por uma expressão em forma de um histograma. A expressão pode ser apenas o nome de uma variável criada dentro do modelo.



Através do painel de edição é possível determinar:

**Expression:** Determina a variável que será monitorada.

**Minimum e Maximum:** Determina os valores máximo e mínimo do elemento gráfico apresentado na tela.

**# Cells:** Define o número de barras verticais a ser apresentado no histograma. O número máximo de células possível é 20.

**Exterior cells:** permite mostrar valores que estejam fora dos valores máximo e mínimo determinados. O número máximo de células externas é 2.

**Cumulative Line:** caso marcada, esta opção permite mostrar ou não a linha com os valores assumidos pela variável em questão ao longo do tempo.

**Border:** Determina o tipo de borda a ser usada no histograma.

**Title:** caso se opte por utilizar um título (opção *Use Title* marcada), é possível editar o texto, sua formatação, e o

posicionamento em relação ao relógio.

É possível, também, configurar a cor através dos botões *Area*, *Border*, *Bar*, *Bar Frame* e *Cumul. Line*. *Transparent Background* diz respeito às características gráficas do objeto (sem cor de fundo).

### 10.2.6. Plot

A ferramenta de animação *Plot* permite mostrar o valor (numérico) de uma ou mais variáveis determinadas por expressões na forma de um gráfico do valor das variáveis ao longo do tempo. A expressão pode ser apenas o nome de uma variável criada dentro do modelo.

Através do painel de edição é possível determinar:

**Expression:** Determina as variáveis que serão monitoradas. Através dos botões paralelos ao campo Expressions é possível editar as variáveis a ser monitoradas.

**Time Range:** Determina o intervalo de tempo a ser apresentado no gráfico.

**Refresh:** Define a forma como será feita a atualização do gráfico a longo do tempo. Caso a opção *Full* esteja selecionada, o gráfico será atualizado quando é completamente preenchido. As opções fracionárias movem o gráfico horizontalmente de acordo com o valor das frações e a opção *None* não atualiza o gráfico uma vez que este tenha sido preenchido.

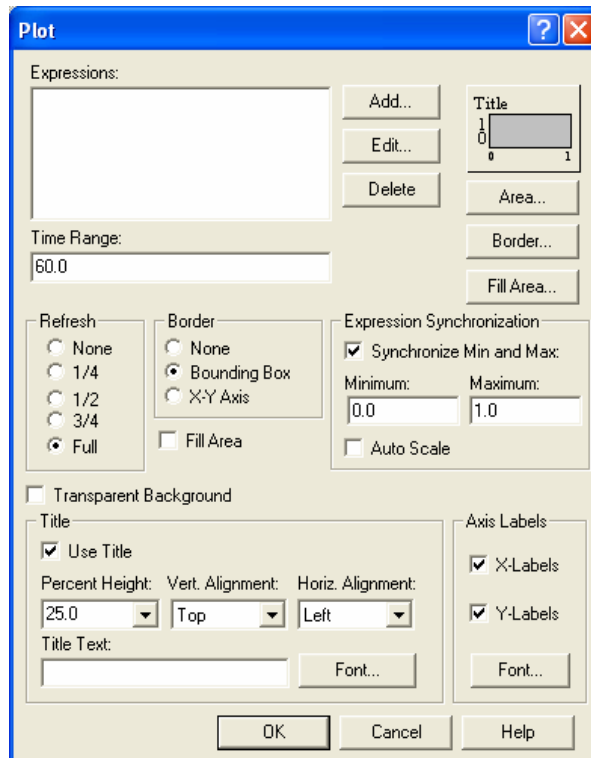
**Border:** Determina o tipo de borda a ser usada no histograma.

**Expression Synchronization:** Caso selecionada, sincroniza os valores máximo e mínimo selecionados para todas as variáveis monitoradas.

**Axis Label:** Permite inserir legendas para os eixos do gráfico;

**Title:** caso se opte por utilizar um título (opção *Use Title* marcada), é possível editar o texto, sua formatação, e o posicionamento em relação ao relógio.

É possível, também, configurar a cor através dos botões *Area*, *Border* e *Fill Area*. *Transparent Background* diz respeito às características gráficas do objeto (sem cor de fundo).

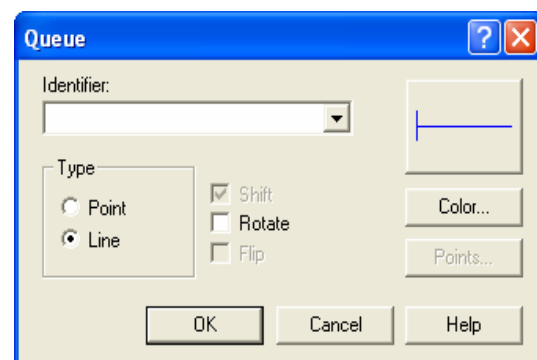


### 10.2.7. Queue

A ferramenta *Queue* permite animar queues definidas no modelo.

No campo *Identifier* deve ser colocado o nome da fila que está sendo animada conforme definido no modelo.

A animação pode ser de dois tipos: *point* ou *line*. Na animação do tipo linha as entidades que vão chegando são enfileiradas uma após a outra. Na animação do tipo *point* as entidades são armazenadas





sobre os pontos definidos (smart 015).

Outras opções são:

**Rotate:** Rotaciona 180 o a figura das entidades que chegam na fila.

**Flip:** Inverte das entidades esperando na fila. Esta opção só está disponível quando é selecionada a opção *Rotate*.

Ainda é possível escolher a cor da linha da animação através do botão color.

### 10.3. Animação de blocos de transferência avançados

É possível criar animações de blocos de transferência avançados. Para tanto, é necessário construir o seu modelo utilizando os blocos que constam no *Advanced Transfer Panel*.

A animação é feita através da barra de ferramentas para animação de transferência (*animate transfer toolbar*).



Figura 72 - Ferramentas para animação de tranferências

Os elementos de animação que constam na barra de ferramentas na ordem em que aparecem são: *storage*, *parking área*, *seize area*, *transporter*, *station*, *intersection*, *route*, *segment*, *distance*, *network*, *promote path*.

#### 10.3.1. Storage

A ferramenta de animação *storage* permite animar *queues* e *storages* definidos no modelo.

No campo *Identifier* deve ser colocado o nome da fila ou storage que está sendo animada conforme definido no modelo.

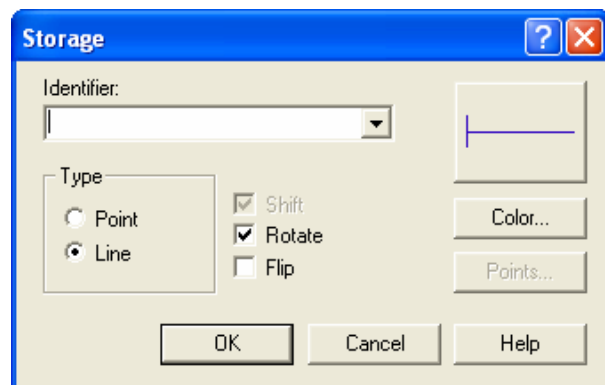
A animação pode ser de dois tipos: *point* ou *line*. Na animação do tipo linha as entidades que vão chegando são enfileiradas uma após a outra. Na animação do tipo point as entidades são armazenadas sobre os pontos definidos (smart 015).

Outras opções são:

**Rotate:** Rotaciona 180 graus a figura das entidade que chegam na fila.

**Flip:** Inverte as entidades esperando na fila. Esta opção só está disponível quando é selecionada a opção *Rotate*.

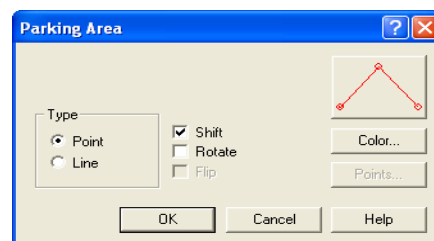
Ainda é possível, escolher a cor da linha da animação através do botão color.



#### 10.3.2. Parking Area

A ferramenta de animação *Parking Area* permite manter *Trasnporters* ou *Resources* que não estejam em transito entre estações. Assim, a imagem associada ao *transporter* poderá estar numa *parking area* somente quando o seu estado for *idle*, *busy* (esperando o processamento de alguma entidade) ou *inactive*.

Uma *parking area* está sempre associada a uma *station*, *intersection* ou *resource*.



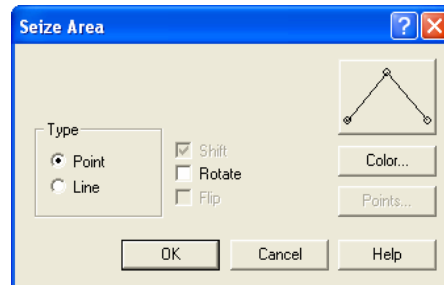


Assim como a ferramenta *storage* uma *parking area* pode ser do tipo *line* ou *point*. As opções *Rotate* e *Flip* têm, também, a mesma função segundo descrito para a ferramenta *storage*. A opção *Shift* não está disponível e é possível escolher a cor da *parking area* pressionando o botão *Color*.

### 10.3.3. Seize Area

A ferramenta de animação *Seize area* permite animar entidades que estão utilizando um recurso ou sendo processadas. Assim sendo, uma *seize area* está sempre associada a um recurso.

As opções de edição disponíveis para essa ferramenta são as mesmas já descritas para a ferramenta de animação *storage*.



### 10.3.4. Transporter

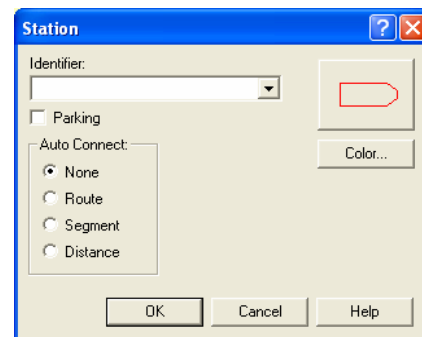
A ferramenta de animação *Transporter* permite animar os transporters incluídos no modelo. A animação de transporters é feita através da fábrica de símbolos.

### 10.3.5. Station

A ferramenta de animação *Station* permite animar as estações constantes no modelo. Ao criar a animação, é necessário que se identifique a estação a qual a animação está associada através do campo *Identifier*.

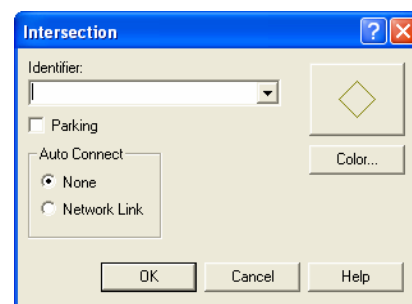
A opção *Parking* determina se a estação terá ou não uma *parking area* associada.

A opção *Auto Connect* permite identificar se haverá ligação entre estações e, também, o tipo de ligação: *Route*, *Segment* ou *Distance*.



### 10.3.6. Intersection

*Intersections* se assemelham a estações na medida em que definem um ponto por onde um transportador deverá passar, podendo ser conectadas através de *networks* de forma a determinar o movimento do transportador. As opções são: Associar ou não uma *parking area* a *intersection*. A opção *auto connect* determina se as estações serão ou não automaticamente conectadas.



### 10.3.7. Route

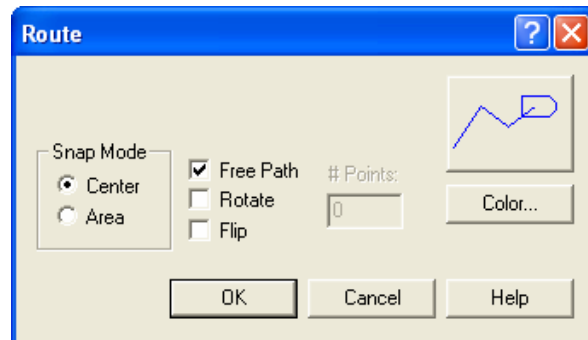
*Routes* são ferramentas de animação do tipo *path* que permitem a animação do bloco de fluxo *Route* definindo caminhos que mostram o movimento do objeto animado no modelo.

As opções de edição são:

**Snap Mode:** Permite definir se o ponto inicial ou final serão no centro da *station* (*Center*) ou em algum ponto da área da *station* definido pelo mouse (*Area*).

**Free Path:** quando selecionada essa opção faz com que *stations*, caso estas não existam, sejam automaticamente adicionadas no início ou final da *route*.

**Rotate e Flip:** Permite definir a orientação da figura conforme descrito no item *storage*.



### 10.3.8. Segment

Ferramenta de animação do tipo *Path* que permitem a animação do bloco de dados *Segment* definindo caminhos que mostram o movimento do objeto animado no modelo. Como se sabe, um conjunto de segmentos formam uma *network*. Além disso, é válido lembrar que *segments* são direcionais.

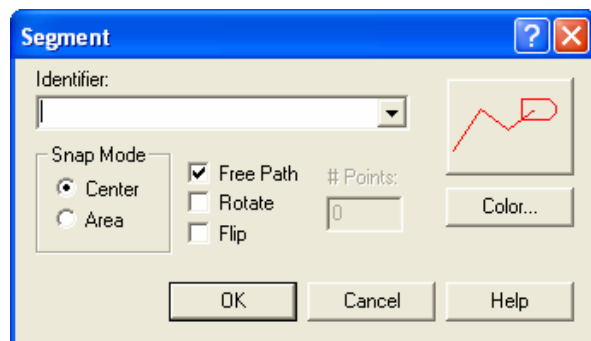
As opções de edição são:

**Identifier:** Numero ou nome que identificam o segmento;

**Snap Mode:** Permite definir se o ponto inicial ou final serão no centro de *station* (*Center*) ou em algum ponto da área da *station* definido pelo mouse (*Area*).

**Free Path:** quando selecionada essa opção faz com que *stations*, caso estas não existam, sejam automaticamente adicionadas no início ou final da *route*.

**Rotate e Flip:** Permite definir a orientação da figura conforme descrito no item *storage*.



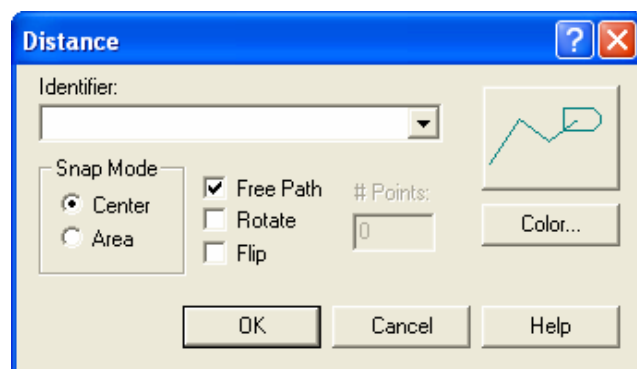
### 10.3.9. Distance

Ferramenta de animação do tipo *path* que permitem a animação do bloco de dados *Distance* através da definição de caminhos que mostram o movimento do objeto animado no modelo.

As opções de edição são:

**Identifier:** Numero ou nome que identificam o segmento;

**Snap Mode:** Permite definir se o ponto inicial ou final serão no centro de *station* (*Center*) ou em algum ponto da área da *station* definido pelo mouse (*Area*).



**Free Path:** quando selecionada essa opção faz com que *stations*, caso estas não existam, sejam automaticamente adicionadas no início ou final da *route*.

**Rotate e Flip:** Permitem definir a orientação da figura conforme descrito no item *storage*.

### 10.3.10. Network

Ferramenta de animação do tipo path que permitem a animação de conjuntos de *Segments* definindo caminhos que mostram o movimento do objeto animado no modelo. Como se sabe, um conjunto de segmentos formam uma *network*. Além disso, é válido lembrar que *segments* são direcionais.

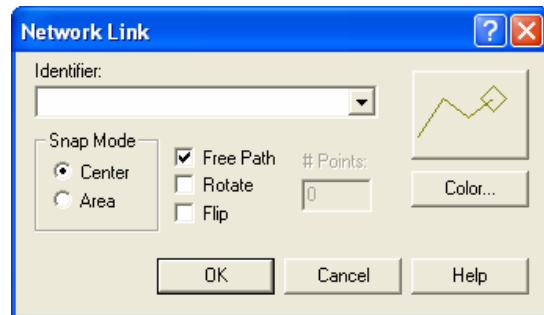
As opções de edição são:

**Identifier:** Número ou nome que identificam o segmento;

**Snap Mode:** Permite definir se o ponto inicial ou final serão no centro de *station* (*Center*) ou em algum ponto da área da *station* definido pelo mouse (*Area*).

**Free Path:** quando selecionada essa opção faz com que *stations*, caso estas não existam, sejam automaticamente adicionadas no início ou final da *route*.

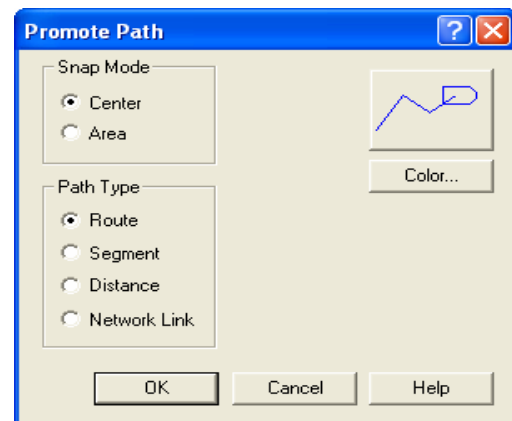
**Rotate e Flip:** Permitem definir a orientação da figura conforme descrito no item *storage*.



### 10.3.11. Promote Path

Ferramenta de animação que permite a transformação de linhas e curvas desenhadas em objetos de animação do tipo *route*, *segment*, *distance* ou *network link*. A opção de edição **Snap Mode**, define se o ponto inicial ou final serão no centro de *station* (*Center*) associada ao path em questão ou em algum ponto da área da *station* definido pelo mouse (*Area*).



Na opção **path type** pode ser escolhido o tipo de path em que a linha ou curva será transformado.

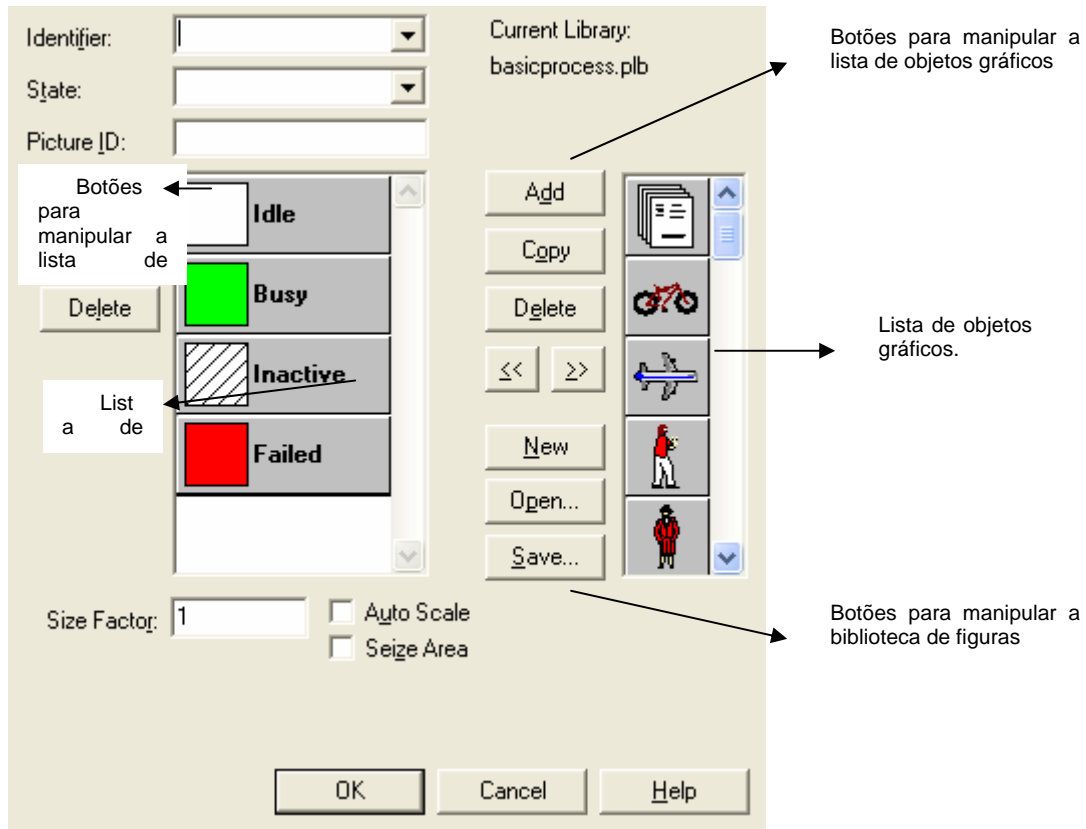


## 10.4. Fábrica de símbolos ARENA

A fábrica de símbolos está disponível para animar *resource*, *transporter* e para animações em geral feitas através do *Global Picture Placement*.


### 10.4.1. Resource e transporter

Acessada através dos ícones  (*resource*) e  (*transporter*) presentes na barra de ferramentas. Ao pressionar o mouse será aberta a tela da Fábrica de símbolos conforme a imagem abaixo:

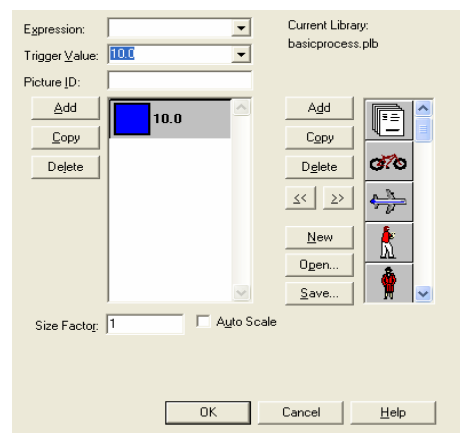


Os campos *Identifier*, *State* e *Picture ID* estão associados as figuras disponíveis. Em *Identifier* deve ser colocado o identificador do recurso a ser animado. É possível associar um objeto gráfico diferente para cada estado definido para o recurso a ser animado. Isso é feito a partir do campo estado. Para cada estado é possível determinar a figura que será representativa do mesmo no campo *PictureID*. A edição de imagens e de figuras e a manipulação da biblioteca são feitas da mesma forma conforme descrito no item 15.1.2 deste manual.

#### 10.4.2. Global Picture Placement

Acesso através do Ícone  presente na barra de ferramentas, a tela da Fábrica de símbolos para *global picture placement* é conforme a imagem ao lado:

Nela estão presentes os mesmos elementos apontados no item anterior, variando apenas os campos relativos à utilização das figuras criadas. Através do Global picture placement pode-se criar uma figura associada a uma expressão que variara de acordo com o resultado da expressão. Para cada picture ID, pode ser exolhido um Trigger Value quando o valor da expressão for igual ao Trigger Value a imagem será alterada para o valor associado Trigger Value.



## **ANOTAÇÕES**