FEMEC - UFU
Uberlândia - MG
2º COBEF
Congresso Brasileiro de Engenharia de Fabricação
ABCM
18 a 21 de Maio de 2003

SIMULAÇÃO DE SISTEMAS DE MANUFATURA ATRAVÉS DA INTERNET

João Carlos Espíndola Ferreira
Universidade Federal de Santa Catarina, Departamento de Engenharia Mecânica, GRIMA/GRUCON, Caixa Postal 476, CEP 88040-900, Florianópolis, SC, jcf@grucon.ufsc.br.

João Paulo Costa e Silva Nunes
Universidade Federal de Santa Catarina, Departamento de Engenharia Mecânica, GRIMA/GRUCON, Caixa Postal 476, CEP 88040-900, Florianópolis, SC, jpcs@matrix.com.br.

Resumo. *O presente trabalho enfoca a implementação de modificações num software de simulação escrito na linguagem Java, denominado JSIM ("Java Simulation"), no intuito de ser utilizado para a realização via Internet de simulações de sistemas de manufatura. Utilizando-se a orientação a objetos, descreve-se a lógica de funcionamento deste software no intuito de concretizá-lo como uma ferramenta de simulação capaz de auxiliar na solução de problemas de manufatura. Neste software a simulação é feita em dois passos: (i) inicialmente o usuário modela o sistema de manufatura, e após terminar-se a modelagem gera-se o código fonte em Java contendo todas as classes referentes a este modelo; (ii) efetua-se a simulação propriamente dita executando-se o código fonte gerado anteriormente, e os resultados obtidos, os quais incluem os dados referentes às filas e às máquinas. Utiliza-se neste sistema o paradigma de simulação de interação de processos, sendo que os resultados estatísticos do modelo simulado podem ser obtidos por mecanismos de simulação em tempo real ou virtual. Para demonstrar a capacidade do software, apresenta-se um exemplo do seu uso na simulação de um sistema simples de manufatura. A disponibilização deste software na Internet tem as seguintes vantagens: (i) qualquer pessoa em qualquer lugar pode executar o software (através de um browser). Ele/ela poderá até ser um engenheiro interessado na modelagem e verificação de seu sistema de manufatura; (ii) este software poderá ser utilizado no ensino de simulação, e os alunos poderão através do software identificar limitações no sistema modelado, e propor meios de melhorar o mesmo.*

Palavras-chave: *Simulação, Internet, Sistema de Manufatura, Orientação a Objetos.*

1. INTRODUÇÃO

No contexto atual de globalização, a competitividade entre as empresas é um fator fundamental no processo de criação e difusão de novas tecnologias. O mercado mundial, mais competitivo e abrangente, exige produtos que agreguem uma ampla gama de qualidades como, preços acessíveis, funcionalidade, praticidade e confiabilidade.

O advento e disseminação das telecomunicações, a melhoria dos transportes em escala mundial integrou mercados ora isolados, estabeleceu novas formas de transações comerciais e permitiu que as informações e conhecimentos gerados experimentassem uma notável flexibilidade e eficiência de propagação. Este dinamismo gerado nos leva a uma conjuntura atual complexa e volátil, onde a informação é um conceito chave mas que normalmente experimenta um período de validade muito curto, pois novos conhecimentos e tecnologias estão constantemente sendo criados e aperfeiçoados. A sobrevivência neste competitivo cenário exige das empresas e profissionais do mercado tecnológico uma grande capacidade de adaptação, além de uma incessante busca por novas metodologias e espírito de vanguarda na concepção de novas idéias e produtos.

O notável desenvolvimento dos computadores, seja em termos de componentes de hardware ou de software, e sua significativa redução de custos, popularizou as ferramentas computacionais nos mais diversos setores. A disseminação da rede mundial de computadores, a Internet, e a possibilidade de utilização de máquinas de grande capacidade de processamento inquestionavelmente vêm alterando abordagens antigas de trabalho e contribuindo com outras novas.

No contexto da fabricação e manufatura, a utilização de técnicas de simulação de sistemas reais de fabricação vem surgindo como uma nova ferramenta de análise e auxílio na tomada de decisões, uma vez que se tem alcançado um nível muito satisfatório de aproximação do comportamento do sistema real a ser simulado nos modelos computacionais. Esta potencialidade de simulação está diretamente associada não somente à evolução dos computadores, mas também à evolução das linguagens de programação, que estão mais práticas e com compiladores mais flexíveis, permitindo assim a construção de modelos de relativa complexidade com códigos fonte mais compactos e eficientes.

A flexibilidade de se construir um modelo computacional para servir de embasamento no estudo de um sistema, e que permita a formulação de perguntas pertinentes sobre o que aconteceria no sistema no caso de uma alteração (ou até mesmo prever o acontecimento de um fato inicialmente não especulado) torna a simulação computacional realmente atrativa, pois é um processo que não gera nenhuma forma de danos e prejuízos, e propicia liberdade para se tentar as mais diversas idéias, incentivando assim a capacidade criativa na solução de problemas de uma maneira que não seria possível trabalhando-se apenas com o sistema real.

Contudo, a construção dos modelos computacionais deve ser cuidadosamente executada, permitindo um bom grau de detalhamento para que as conclusões tiradas sobre o modelo se aproximem razoavelmente das que seriam extraídas na execução do próprio sistema físico real (Filho, 1995).

2. O SOFTWARE JSIM

O software JSIM (http://orion.cs.uga.edu:5080/~jam/jsim) é um ambiente de simulação e animação desenvolvido na Universidade da Geórgia utilizando a linguagem Java para o seu desenvolvimento (Sun, 2000). Trata-se de um programa ainda em desenvolvimento, gratuito, com código fonte aberto que se utiliza das potencialidades da linguagem Java para a execução dinâmica de simulações na rede mundial de computadores. No intuito de adequa-lo como um software simulador disponível para execução de simulações de sistemas de manufatura, implementações adicionais e adaptações de classes previamente existentes foram executadas, tendo como enfoque principal a idéia de uma situação onde um usuário acessa remotamente o software através da Internet para execução de modelos de simulação.

Basicamente, o JSIM é um programa simulador que disponibiliza uma ampla biblioteca de classes Java, no intuito do desenvolvimento de modelos simples de simulação (Miller et al., 1997, 1998). Java é uma poderosa e elegante linguagem de programação orientada a objetos com vários recursos inclusos muito úteis no desenvolvimento de simulações, que permite que modelos de simulações fiquem disponíveis na Internet. Por ser uma linguagem de programação orientada a objetos bem estruturada, Java facilita a extensa reutilização de classes previamente elaboradas (Horstmann e Cornel, 2001a, 2001b). Essa linguagem fornece recursos para construção de ambientes dinâmicos e com animações, sendo portanto melhor do que os ambientes antigos de simulação, estáticos, apenas com recursos textuais.

Como Java é uma linguagem que independe da plataforma, podendo ser executada em qualquer sistema operacional, ela permite um grande dinamismo aos aplicativos de rede (Chan et al, 2000). A possibilidade de execução de *applets* (programas aplicativos em Java) em praticamente qualquer navegador e o suporte que a linguagem oferece a programação em múltiplas linhas de execução (*threads*) (Horstmann e Cornell, 2001b) dão suporte à implementação dos processos de interação de entidades e agendamento de eventos necessários à execução de simulações.

A biblioteca do JSIM oferece pacotes de classes para duas formas de abordagem de simulações, que são: a abordagem de agendamento de eventos e a de interação de processos. A abordagem de agendamento de eventos está centrada na concepção de uma simulação como sendo uma seqüência rigidamente determinada de eventos a acontecer (Kelton et al, 1998). Há total controle de onde os eventos ocorrem, e o

que acontece depois deles ocorrerem. Existe uma grande flexibilidade com respeito a atributos e variáveis, e é possível saber o estado de qualquer coisa em qualquer momento. Esta abordagem já foi largamente utilizada no passado, e apesar de ser aparentemente simples e vantajosa, torna-se bastante complexa para modelos longos com vários tipos diferentes de eventos e entidades.

Uma abordagem mais natural para simulações seria tomar o ponto de vista de uma entidade típica de simulação e seguir processando como ela trabalha e se comporta ao longo do modelo, ao invés da orientação onisciente de um controlador de eventos gerenciando todas entidades, atributos, variáveis e acumuladores estatísticos como é feita no paradigma de agendamento de eventos. Na concepção da interação de processos o ponto central está nos processos pelos quais as entidades passam (Kelton et al,1998). Por apresentar uma concepção mais flexível e dinâmica, a lógica de interação de processos hoje é mais utilizada, no entanto ambas as formas de abordagem podem ser trabalhadas com o JSIM. O exemplo de simulação de um sistema simples de manufatura a ser apresentado utilizará a abordagem de interação de processos.

2.1. Estrutura

O JSIM é composto de três módulos básicos. O primeiro é uma *applet* que atua como um construtor gráfico de modelos capaz de trabalhar com um vetor flexível de classes utilizado para a construção inicial de modelos. A idéia básica é que o construtor gráfico de modelos gere uma representação estática de modelos de simulação. Isto é feito utilizando-se representações gráficas básicas com nós e elementos de ligação entre os nós. Existem diversos tipos de nó, e cada nó realiza uma tarefa específica, como gerar ou eliminar entidades, promover atrasos ou adiantamentos entre as entidades. Os elementos de ligação permitem o fluxo de entidades entre os nós, estabelecendo a dinâmica do modelo. A figura 1 ilustra um exemplo da construção de um modelo de simulação utilizando o construtor gráfico. Observa-se a colocação dos nós, a configuração de suas propriedades na caixa de diálogo e o estabelecimento da ligação entre os nós.

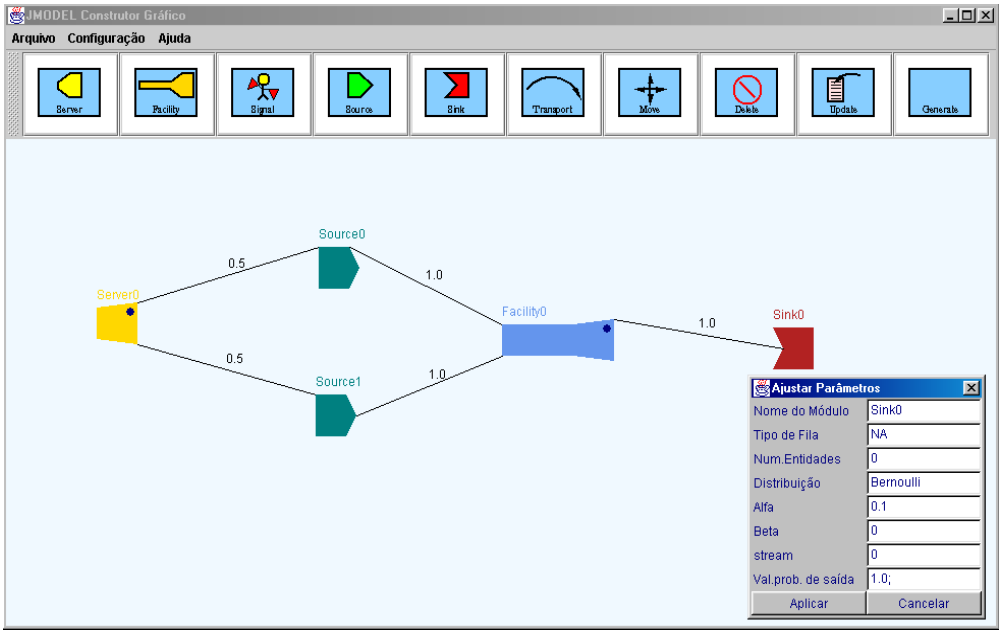


Figura 1. Exemplo da construção de um modelo de simulação utilizando o construtor gráfico

De posse de um modelo básico construído, há a necessidade de salvar este para a posterior geração de um arquivo em Java (*".java")* que será o código fonte do modelo de simulação. O software possui classes específicas que propiciam o recurso de geração automática de um código fonte a partir dos elementos

inseridos no modelo gráfico. Tem-se assim um arquivo *.java* que representa uma simulação específica, bastando então compilar este arquivo *.java*, resultando no arquivo compilado "*.class*" pronto para executar o modelo construído.

Após esta etapa utiliza-se o segundo módulo, a *applet* de execução. Esta *applet* permite a execução dinâmica de simulações, utilizando-se de recursos de animações gráficas que conferem uma maior clareza ao processo de simulação em execução. A lógica de funcionamento da *applet* de execução baseia-se na sua comunicação com o terceiro módulo, a *applet* de controle, que lhe envia os parâmetros inseridos pelo usuário para posteriormente serem criados vários objetos (instâncias de classes) que realizam a animação do modelo usando métodos implementados específicos para tais fins.

No intuito de melhorar o *layout* das interfaces tanto do construtor gráfico de simulações quanto da *applet* de execução dos modelos, trabalhou-se com o objetivo de deixá-las mais informativas ao usuário e com uma interface mais amigável. Na *applet* de execução, dois novos relógios foram implementados na classe, um para cada tipo de simulação, em tempo real ou virtual. Um conjunto adicional de informações foi disponibilizado na tela de execução além do tempo de simulação, para deixar o usuário com mais informações sobre o modelo que ele está executando (figura 2). Ter *applets* de execução bastante informativas é um fator importante quando são executadas várias replicações de simulações diferentes, pois de contrário pode haver confusão por parte do usuário sobre qual modelo ele está visualizando e dificuldades para saber qual é tabela de saída de dados relativa à simulação a partir da qual ele desejava obter informações.

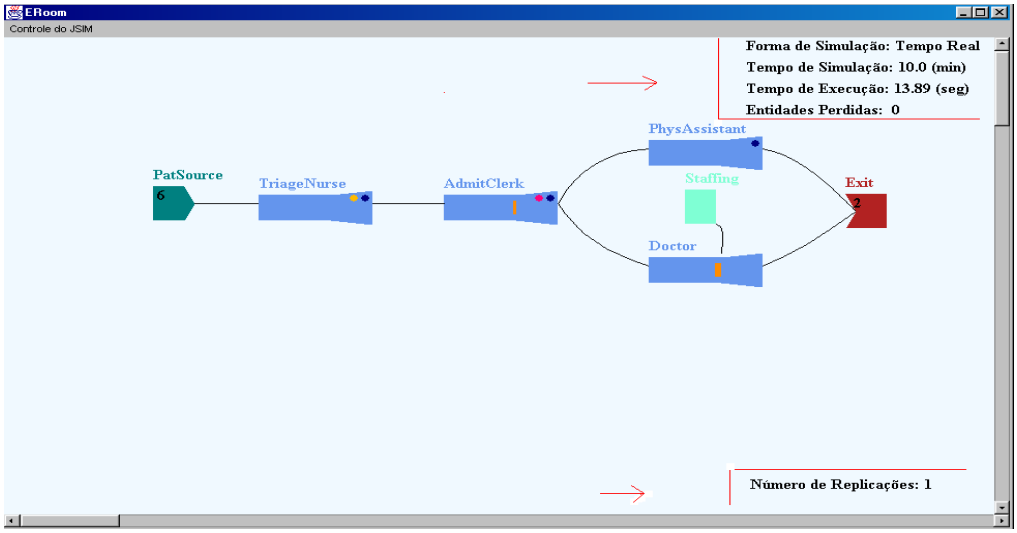


Figura 2. *Applet* de execução de uma simulação

Acrescentou-se informações sobre a forma de simulação, tempo de simulação, tempo de execução com o relógio vinculado à forma de simulação escolhida, contador de entidades aleatoriamente perdidas no processo de execução e número de replicações do modelo.

O terceiro módulo do JSIM, denominado *applet* de controle (figura 3), foi implementado para que o usuário entre com os dados, além de permitir o gerenciamento das simulações anteriormente criadas e que já estão com seus arquivos *.class* prontos para a execução. Na *applet* de controle o usuário pode selecionar uma simulação, bem como atribuir parâmetros como tempo de simulação, número de replicações a serem executadas, número máximo de entidades a serem criadas, forma de simulação, seja em tempo real ou virtual. Na concepção original do JSIM, todos os parâmetros tinham de ser inseridos nas classes do código fonte de uma maneira bastante dispersa, sendo difícil para um usuário simples identificar onde deveriam ser inseridos os parâmetros anteriormente citados.

A *applet* de controle realiza o trabalho de ligação entre os dois módulos iniciais. Por exemplo, considere uma situação onde o usuário execute uma simulação através da *applet* de controle, e após a

observação dos resultados estatísticos ele deseje alterar a distribuição estatística de um ou mais nós no modelo desta simulação no intuito de avaliar o efeito dessas alterações no comportamento do sistema que está sendo simulado. Nesta situação, o usuário deverá acessar o construtor gráfico para efetuar as modificações necessárias e posteriormente o construtor gráfico criará um o novo arquivo *.java* que comporta estas alterações. Sendo este arquivo compilado, tem-se um arquivo *.class* necessário para a execução do modelo de simulação.

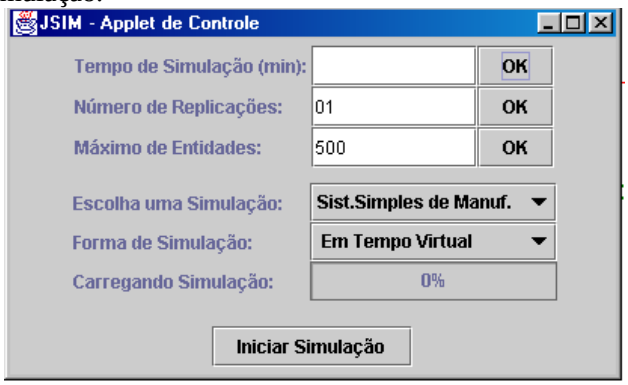


Figura 3. A *applet* de controle

Ao final de todo o processo gera-se uma tabela de dados estatísticos com os dados pertinentes à simulação. Alguns exemplos destes dados serão mostrados no final deste artigo.

2.2. A biblioteca de classes do JSIM

O JSIM apresenta uma biblioteca de classes que tem por objetivo facilitar a construção de modelos mais complexos de simulação (Miller et al, 1997). Os pacotes de classes disponíveis permitem a construção de modelos com códigos fontes reduzidos. A biblioteca é dividida basicamente em cinco pacotes:

Queue – Pacote de filas que inclui quatro diferentes tipos: FIFO (*First In-First Out*), LIFO (*Last In-First Out*), *Priority Queue*, *Temporal Queue*.

A maioria dos métodos do pacote de filas é abstrato, ou seja, estão por ser implementados em subclasses caso venham a ser utilizadas numa simulação específica.

Statistic - O pacote estatístico contém classes para a coleta de informações estatísticas. A classe *Statistic* é uma classe abstrata que contém métodos para analisar dados estatísticos e auxiliar no processo de retorno deles na tabela final de saída de dados.

Existem três classes, *SampleStat*, *TimeStat* e *BatchStat* que estendem a classe básica *Statistic*. A classe *SampleStat* é usada para a coleta de dados estatísticos de exemplares isolados. *TimeStat* é usada para coleta de dados relativos ao tempo, seja para um grupo de entidades ou apenas para um exemplar, registrando o tempo de permanência numa fila ou tempo de processamento e espera num determinado nó. Já *BatchStat* é utilizada na análise de grupos ou lotes de entidades, visando saber o tamanho de um lote, ou conjuntamente com *TimeStat* saber o tempo de permanência de um lote de entidades num nó, ou tempo de espera de um lote em uma fila qualquer.

Os métodos da classe *Statistic* lhe dão potencialidade para cálculos de mínimos, máximos, médias, variâncias, desvio padrão e outros parâmetros estatísticos.

Variate- Este pacote propicia a capacidade de se gerar distribuições aleatórias. Basicamente o pacote apresenta a implementação de diversas distribuições estatísticas usualmente utilizadas, como a distribuição de Bernoulli, normal, binomial, Poisson, qui-quadrado, exponencial, *t* de Student, entre outras.

Event - É o pacote disponibilizado para a construção de simulações através do agendamento de eventos. Este pacote é composto das classes *Event*, *Entity* e *Scheduler*. O agendamento dos eventos nas simulações usa uma lista de eventos futuros gerenciada pelas três classes anteriores.

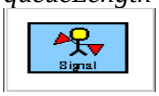
Process - Esse pacote provê classes utilizadas quando cria-se modelos de simulação através da interação de processos. Considerando-se que uma simulação é composta de nós e elementos de transporte, tem-se então neste pacote classes que representam estes diversos tipos de nós como:

Source – Nó gerador de entidades na simulação, dependendo dos parâmetros inseridos no construtor da classe, como tempo de espera entre a criação de uma entidade e a anterior, ou número máximo de entidades a serem criadas. *Source* é uma classe abstrata, pois diferentes tipos de simulação requerem diferentes tipos de entidades a serem criadas. Desta forma, cada simulação necessita de uma implementação diferente dos métodos da classe *Source*.

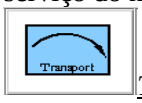
Sink – Nó consumidor de entidades na simulação que basicamente destrói as entidades que chegam e automaticamente coleta dados estatísticos referentes às entidades destruídas. O nó *Sink* realiza exatamente o oposto do nó *Source*, e as classes relativas a estes nós têm a mesma concepção de funcionamento.

Server - Nó que provê serviços para as entidades. Inicialmente, na construção dos modelos de simulação, define-se o número de nós prestadores de serviço que se deseja, e então as entidades obtêm alguma forma de serviço solicitando um nó *Server*, utilizando-o e posteriormente liberando-o. Este tipo de nó também possui a funcionalidade de coleta de dados estatísticos, principalmente referentes ao tempo em que um determinado nó esteve prestando um serviço.

Facility - Nó que tem suas características derivadas do nó *Server*. É basicamente um nó que também provê serviços, constituindo-se de um grupo de unidades que presta serviços alimentados por uma fila em comum. Este nó encapsula as classes de gerenciamento de filas e de transporte de entidades para poder executar tais tarefas. Para obter um serviço deste nó, novamente as entidades solicitam o serviço. A diferença para o nó *Server* é que caso o nó *Facility* esteja ocupado, automaticamente o gerenciador de filas do nó, utilizando o método *enqueue*, coloca a entidade numa fila de espera. Quando a entidade acaba de receber o serviço prestado pelo nó, ela libera o nó utilizando-se um método *release* de liberação. O método *queueLength* retorna o tamanho da fila gerada no nó *Facility*.



Signal - Nó auxiliar que apenas altera através de sinais a quantidade de unidades de serviço que um nó *Server* ou *Facility* pode ter para prestar serviços para as entidades. Um nó do tipo *Signal* altera o comportamento de nós que prestam serviço, aumentando ou diminuindo o número de unidades de serviço do nó.



Transport - Elemento de transporte para conexão dos nós. Uma entidade deixando um nó escolhe de maneira probabilística um dos caminhos para deixar o nó e encaminhar-se para outro, caso exista a opção de mais de um caminho.

O pacote *process* também possui classes que realizam o gerenciamento do tempo nas simulações. É possível, em função do tempo, a execução de dois tipos de simulação: a simulação em tempo virtual e aquela em tempo pseudo-real.

A simulação em tempo virtual é recomendada para grandes intervalos de simulação, quando o usuário não poderia esperar o tempo real equivalente, ou quando a simulação apresenta elevado custo computacional. Um relógio virtual é usado ao invés de um relógio real. A classe *Clock* controla o tempo tanto para simulações de tempo pseudo-reais ou virtuais, sendo que o mecanismo de avanço virtual do tempo no caso de simulações virtuais é processado pela classe *VirtualScheduler*.

3. EXECUÇÃO DE UM MODELO DE SIMULAÇÃO

O modelo elaborado simula um sistema simples de manufatura, com um depósito de peças vinculado a uma esteira que encaminha as peças para uma máquina que executa uma determinada operação nestas peças. Apenas uma peça é trabalhada por vez na máquina. Após a execução da operação, a peça trabalhada é encaminhada para uma outra esteira e é levada para um armazém de materiais. A simulação foi executada em tempo real e virtual, com um valor de 15 minutos.

A lógica de simulação deste exemplo, pela concepção da interação de processos, deve ser concebida colocando-se uma peça como uma entidade que irá seguir o processo descrito anteriormente. A lógica do modelo na concepção de uma entidade é a seguinte:

- Criar a entidade (uma nova entidade chega num nó do tipo *Source*).
- Marcar o tempo agora, caso seja esse um dos seus atributos, para posterior coleta de dados de chegada e para cálculos futuros de tempo em fila e tempo em processamento.
- Entrar na esteira, colocando-se no fim da fila (obviamente a primeira entidade dirige-se diretamente para a máquina).
- Esperar na fila até a máquina estar liberada (nó do tipo *Facility*).
- Entrar na máquina para receber a operação, saindo da fila.
- Calcular o tempo na fila, e efetuar o seu registro.
- Permanecer na máquina uma quantidade de tempo igual à requerida como tempo de processamento.
- Após o tempo de processamento, liberar a máquina para que outras entidades possam entrar.
- Incrementar o acumulador de peças produzidas, e calcular o seu tempo de processamento.
- Encaminhar a entidade para o armazém de materiais (nó do tipo *Sink*), e eliminar a entidade.

Utilizando-se o construtor gráfico, cria-se um modelo estático da simulação com os seguintes passos (figura 4):

- Um nó do tipo *Source*, representando o depósito de peças. O máximo de entidades que o nó pode criar são quinhentas unidades. Para representar a saída aleatória de entidades do nó, usou-se uma distribuição exponencial com média de cinco minutos.
- Um nó do tipo *Facility* representando a máquina de processamento. O processamento será representado por uma distribuição triangular, com um mínimo de um minuto, máximo de oito minutos e modo de 4 minutos. A fila que se estabelece antes do nó *Facility* é do tipo FIFO (*first in first out*, ou seja, primeira entidade a entrar na fila sai, automaticamente uma nova entidade entra no fim da fila).
- Um nó do tipo *Sink*, representando o armazém de materiais, com a mesma distribuição exponencial do nó *Source*.

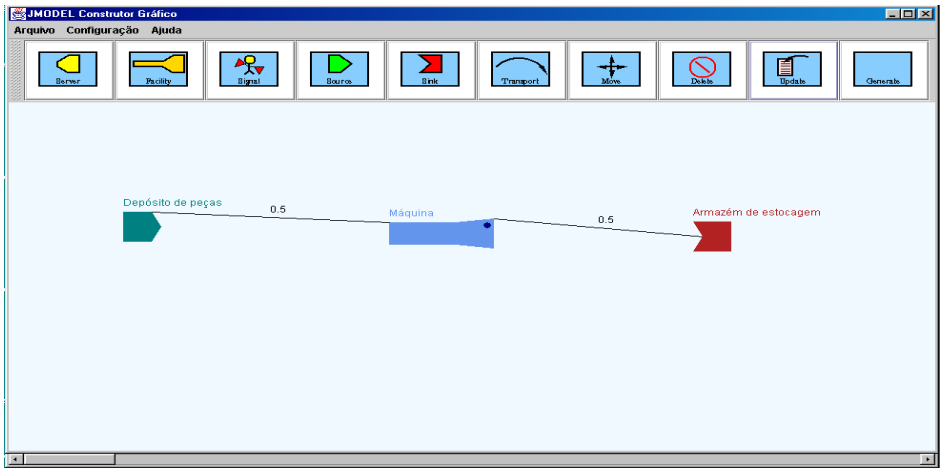


Figura 4. Modelo estático do sistema simples de manufatura.

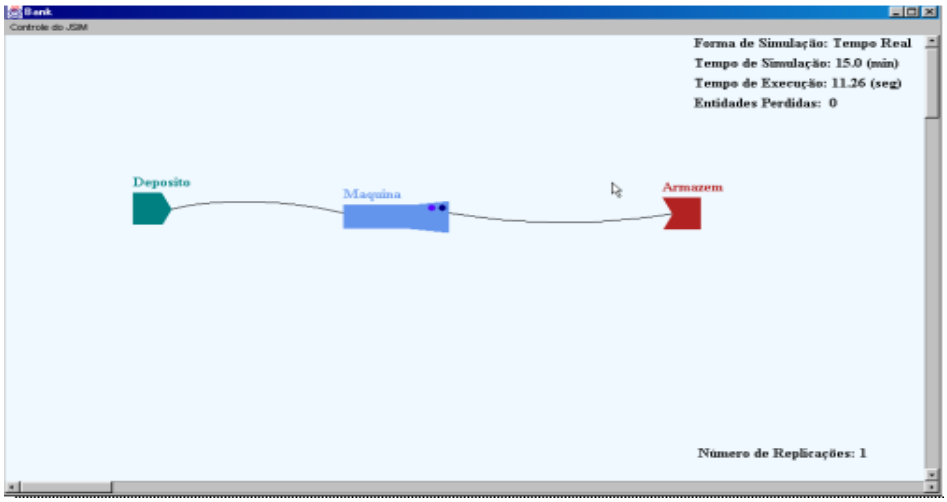


Figura 5. Execução da simulação

Uma janela da execução do modelo é ilustrada na figura 5, e as tabelas de resultados são mostradas nas tabelas 1 e 2. A seguir descreve-se como a tabela 1 é interpretada.

Tabela 1. Tabela de resultados para o modelo executado em tempo real

Janela de Resultados Estatísticos

Nome do Módulo	Num.Entidades	Valor.Min	Valor.Max	Valor Médio
Deposito (Duração)	0.0	0.0	0.0	0.0
-Deposito (Duração)	2.0	529.21	542.71	535.96
Deposito (Ocupação)	0.0	0.0	0.0	0.0
MaquinaQ(Duração)	0.0	0.0	0.0	0.0
-MaquinaQ(Duração)	2.0	0.0	0.0	0.0
MaquinaQ(Ocupação)	0.0	0.0	0.0	0.0
Maquina (Duração)	0.0	0.0	0.0	0.0
-Maquina (Duração)	2.0	228.17	284.08	256.12
Maquina (Ocupação)	12.647	0.0	1.0	0.6750
Armazem (Duração)	0.0	0.0	0.0	0.0
-Armazem (Duração)	2.0	230.57	286.55	258.56
Armazem (Ocupação)	0.0	0.0	0.0	0.0

Tabela 2. Tabela de dados para modelo executado em tempo virtual

Janela de Resultados Estatísticos

Nome do Módulo	Num.Entidades	Valor.Min	Valor.Max	Valor Médio
Deposito (Duração)	0.0	0.0	0.0	0.0
-Deposito (Duração)	2.0	529.24	542.80	536.02
Deposito (Ocupação)	0.0	0.0	0.0	0.0
MaquinaQ(Duração)	0.0	0.0	0.0	0.0
-MaquinaQ(Duração)	2.0	0.0	0.0	0.0
MaquinaQ(Ocupação)	0.0	0.0	0.0	0.0
Maquina (Duração)	0.0	0.0	0.0	0.0
-Maquina (Duração)	2.0	228.22	284.14	256.18
Maquina (Ocupação)	12.641	0.0	1.0	40531.
Armazem (Duração)	0.0	0.0	0.0	0.0
-Armazem (Duração)	2.0	230.62	286.54	258.58
Armazem (Ocupação)	0.0	0.0	0.0	0.0

- Terceira linha - Observa-se que duas peças estiveram no depósito, e a que ficou menor tempo no depósito lá permaneceu por 529.21 segundos (8.82 minutos), e a que ficou o maior tempo permaneceu 542.21 segundos (9.03 minutos). O valor médio de permanência das peças foi de 535.96 segundos (8.93 minutos).
- Sexta linha - Das duas peças provenientes do depósito, ambas passaram pela fila na esteira da máquina (*MáquinaQ* na tabela) e que não houve atrasos na fila com as peças dirigindo-se diretamente à máquina, uma vez que os tempos mínimos e máximos de permanência na fila são nulos (colunas três e quatro da linha em questão).
- Nona linha - As duas peças envidas pela esteira para a máquina foram trabalhadas, e a peça que teve o menor tempo de manufatura foi processada em 228.17 segundos (3.80 minutos), enquanto que o processamento mais demorado ocorreu em 284.08 segundos (4.73 minutos). Com isso o tempo médio de processamento das duas peças foi de 256.12 segundos (4.27 minutos).
- Décima linha - Esta linha fornece informações a respeito do tempo total em que a máquina ficou ocupada. Na segunda coluna da referida linha, observa-se o valor de 12.65 minutos para o tempo total que a máquina atuou durante a execução do modelo. Confirma-se na quarta coluna desta linha que realmente apenas uma peça foi manufaturada por vez, de acordo com o proposto originalmente. A capacidade de trabalho da máquina é um parâmetro que pode ser facilmente alterado e que exerce grande influência nos resultados.
- Décima segunda linha - Das duas peças fabricadas, ambas foram encaminhadas para o armazém de materiais, e portanto nenhuma peça terminou o tempo de simulação em trânsito nas esteiras. Dentre estas peças, a que permaneceu menos no armazém ficou 230.57 segundos (3.84 minutos), e a de maior tempo de permanência ficou (4.78 minutos), concluindo-se assim que o tempo médio de permanência das peças no armazém de entidades foi de 4.31 minutos, e após este tempo médio as entidades que representam as peças foram extintas, simbolizando a retirada das peças do armazém de materiais.

Executou-se o mesmo modelo, com as mesmas configurações da execução anterior utilizando mecanismos de simulação virtual e os resultados foram bastantes similares. Além disso, utilizou-se o software comercial Arena (Kelton et al, 1998) para comparação dos resultados e observou-se novamente equivalência nos dados resultantes.

O link para a execução deste software pode ser encontrado na página http://www.grima.ufsc.br.

4. CONCLUSÕES

Neste artigo foi apresentado um software de simulação desenvolvido na linguagem Java, que pode ser executado por qualquer pessoa em qualquer lugar do mundo. Os resultados obtidos por este software foram comparados com aqueles obtidos pelo software comercial Arena para o mesmo modelo, e os mesmos foram equivalentes. O software fornece um módulo para a modelagem de sistemas a eventos discretos, e o usuário pode assim considerar diferentes problemas e situações. Como mencionado anteriormente, uma das vantagens deste software é que, por ser gratuito e estar disponível na Internet, escolas e universidades em qualquer lugar podem utilizá-lo para o ensino de simulação a eventos discretos.

Como trabalho futuro, pretende-se desenvolver classes que permitam a tomada de decisões pelo software de simulação quanto à seqüência de eventos futuros a serem executados no controle de um sistema de manufatura real.

5. AGRADECIMENTOS

Os autores agradecem ao CNPq pelo suporte financeiro às atividades deste projeto.

6. REFÊRENCIAS

Chan, M.C., Griffith, S.W., Iasi, A.F., 2000, "Java - 1001 Dicas de Programação", Makron Books, São Paulo, SP.

Filho, C.P., 1995, "Introdução à Simulação de Sistemas", Ed. da UNICAMP, Campinas, SP.

Horstmann, C.S., Cornell, G., 2001, "Core Java 2 – Volume I – Fundamentos", Makron Books, São Paulo, SP.

Horstmann, C.S., Cornell, G., 2001, "Core Java 2 – Volume II – Recursos Avançados", Makron Books, São Paulo, SP.

Kelton, W.D., Sadowski, R.P. e Sadowski, D.A., 1998, "Simulation with Arena", McGraw-Hill, New York.

Miller J.N., Zhang, Z. e Zhao, H., 1997, "JSIM: A Java-Based Simulation and Animation Environment", The 30th Annual Simulation Symposium, pp. 31-42, Atlanta.

Miller J.N., Ge, Y. e Tao J., 1998, "Component Based Simulation Environments: JSIM as a Case of Study Using Java Beans", Winter Simulation Conference.

Sun Microsystems, 2000, "The Source for Java™ Technology", http://java.sun.com

SIMULATION OF MANUFACTURING SYSTEMS THROUGH THE INTERNET

João Carlos Espíndola Ferreira
Universidade Federal de Santa Catarina, Departamento de Engenharia Mecânica, GRIMA/GRUCON, Caixa Postal 476, CEP 88040-900, Florianópolis, SC, jcf@grucon.ufsc.br.

João Paulo Costa e Silva Nunes
Universidade Federal de Santa Catarina, Departamento de Engenharia Mecânica, GRIMA/GRUCON, Caixa Postal 476, CEP 88040-900, Florianópolis, SC, jpcs@matrix.com.br.

Abstract. *This paper focuses on the implementation of modifications in a simulation software written in the Java language, called JSIM ("Java Simulation"), so that it is used to perform simulations of manufacturing systems through Internet. By using object orientation, the operation logic of this software is described so that it is used to support the solution of manufacturing problems. In this software simulation is performed in two steps: (i) initially the user models the manufacturing system, and after finishing the modeling process the Java source code is generated containing all the classes in the model; (ii) simulation itself is performed by executing the code source generated previously, and the results are thus obtained, and they include the data regarding the queues and the machines. In this software it is used the paradigm of simulation by process interaction, and the statistical results of the simulated model can be obtained by simulation mechanisms in real or virtual time. To demonstrate the capability of the software, an example of its use in the simulation of a simple manufacturing system is presented. By enabling the use of this software through the Internet, the following advantages result: (i) anyone anywhere can execute the software (through a browser). This person may be an engineer who is interested in modeling and checking his manufacturing system; (ii) this software can be used for teaching simulation, and the students will be able through the software to identify limitations in the modeled system, and to propose means of improving it.*

Keywords: *Simulation, Internet, Manufacturing System, Object Orientation.*