USO DA REALIDADE VIRTUAL PARA ENSINO DE AUTOMAÇÃO DA MANUFATURA

Jeferson Luiz Curzel[1], Marcelo da Silva Hounsell[2], André Bittencourt Leal[3]

Abstract *— In this article the development of a virtual environment for visualization of the functioning of a manufacture cell will be boarded. It will be presented the cell of manufacture and its functioning sequence, as well as the necessary elements for the development of the virtual environment in VRML. In general the equipment used in laboratory is expensive, being used an only cell of manufacture to develop the activities, what it can compromise the learning. With the use of the Virtual Reality each student can carry through the experiment in the virtual environment and after solved the problem he is that the real implementation will be made. Thus, without having a previous knowledge of the programming of the devices of the manufacture cell, the student can visualize and interact with the virtual equipment of form to verify what movement is possible, determination of trajectories and solution of collision problems or blockade.*

Index Terms — Discrete Events, Manufacturing Cell, Virtual Reality.

INTRODUÇÃO

A utilização da Realidade Virtual (RV) nas diversas áreas do conhecimento está em crescente expansão. Esse crescimento é devido a vários fatores, dentre os quais podemos citar, por exemplo, os estudos envolvendo RV nas universidades e a sua respectiva divulgação nas indústrias, o barateamento das interfaces e o acesso cada vez maior das pessoas à informática.

Na educação e treinamento, a RV tem potencial pois propicia o processo de exploração, descoberta, observação e construção de uma nova visão do conhecimento, oferecendo ao aprendiz a oportunidade de melhor compreensão do objeto de estudo, proporcionando não apenas a teoria, mas também a experimentação prática, com o uso de laboratórios didáticos [1].

Assim, com a RV é possível adquirir conhecimentos e realizar treinamento em equipamentos sem, necessariamente, estar fisicamente presente ou com o dispositivo real para utilização. Isso permite que a tarefa possa ser realizada diversas vezes, sem comprometer a segurança pessoal nem provocar danos materiais.

Além disso, mesmo sem ter um conhecimento prévio da programação ou do funcionamento dos dispositivos que compõem uma célula de manufatura, o usuário (aluno) pode visualizar e interagir com os equipamentos virtuais de forma a verificar o que é possível ser feito em termos de execução de movimentos, determinação de trajetórias e problemas com colisão ou bloqueio (situação em que o sistema não completa uma seqüência).

RV: CONCEITOS BÁSICOS

Existem muitas definições de RV envolvendo aspectos gerais ou conceitos tecnológicos. Há várias definições aceitas, o que é devido em parte, à natureza interdisciplinar da área e também à sua evolução a partir dos sistemas computacionais de mesa, simuladores e sistemas de tele-operação [1].

Latta e Oberg [2] citam RV como uma avançada interface homem-máquina que simula um ambiente realista e permite que participantes interajam com ele. Pimentel e Teixeira [3] definem RV como o uso da alta tecnologia para convencer o usuário de que ele está em outra realidade - um novo meio de "estar" e "tocar" em informações. Em termos conceituais, RV é uma realidade que é aceita como verdadeira, embora não necessariamente exista fisicamente [4]. O termo RV é bastante abrangente e acadêmicos, desenvolvedores de *software* e pesquisadores tendem a defini-lo com base em suas próprias experiências. Ele é um paradigma pelo qual se usa um computador para interagir com algo que não é necessariamente físico, mas que pode ser considerado real enquanto está sendo usado [5].

Uma interface de RV torna possível ao usuário navegar em um mundo virtual, interagindo com a aplicação em tempo real. Assim, o usuário sente que faz parte do ambiente que o envolve, gerando o sentimento de imersão. Em muitos casos a RV só precisa **parecer** real, não necessariamente **ser** real [6]. Há duas divisões principais para os sistemas de RV: Realidade Virtual Imersiva (RVI) e Realidade Virtual Não Imersiva (RVNI) [1]. A RVI implica no uso de dispositivos que ocultem o mundo real, como o capacete de visualização, o qual possibilita ao usuário visualizar apenas o mundo virtual. Por outro lado, na RVNI o usuário pode utilizar um monitor convencional para visualizar o mundo virtual e o *mouse* para interagir com o ambiente.

O dimensionamento de um sistema de RV vai depender da sua finalidade. Os níveis de imersão e interatividade são determinados basicamente, de acordo com os dispositivos utilizados para entrada e saída de dados do sistema, bem como a velocidade para processamento em tempo real do computador que executa o sistema de RV [7].

1 Jeferson Luiz Curzel, Sociedade Educacional de Santa Catarina – Instituto Superior Tupy, Rua Albano Schimidt, 3333, Bairro Boa Vista, 89227-700, Joinville, SC, Brasil, jeferson.curzel@sociesc.org.br
2 Marcelo da Silva Hounsell, Universidade do Estado de Santa Catarina, Departamento de Ciência da Computação, Campus Universitário Prof. Avelino Marcante s/n, Bairro Bom Retiro, 89223-100, Joinville, SC, Brasil, marcelo@joinville.udesc.br
3 André Bittencourt Leal, Universidade do Estado de Santa Catarina, Departamento de Engenharia Elétrica, Campus Universitário Prof. Avelino Marcante s/n, Bairro Bom Retiro, 89223-100, Joinville, SC, Brasil, leal@joinville.udesc.br

SISTEMAS DE AUTOMAÇÃO E EVENTOS DISCRETOS

Num contexto abrangente, sistemas são conjuntos de elementos (subsistemas) entre os quais se pode definir uma relação de estados e eventos que operam com uma estrutura organizada [8]. O comportamento de um sistema depende do conjunto de estados e dos eventos de cada subsistema, sendo que em um mesmo estado de tempo podem ocorrer mudanças em vários subsistemas.

Os sistemas de automação construídos pelo homem são denominados Sistemas Dinâmicos a Eventos Discretos, ou Sistemas a Eventos Discretos (SEDs) e sua modelagem matemática pode ser feita pela Teoria de Controle Supervisório [9]. O principal objetivo do controle supervisório é a coordenação desses subsistemas de forma que atendam a uma série de tarefas individuais e conjuntas, garantindo um bom funcionamento global do sistema [10].

Sistemas a Eventos Discretos (SEDs)

Um SED é definido como sendo um sistema dinâmico cuja evolução de estado depende da ocorrência de eventos. Há três características básicas envolvidas na definição de um SED, que são: [8]

- O ciclo de funcionamento é descrito através da seqüência de eventos que determinam a execução ou o final de uma tarefa;
- A ocorrência de eventos simultâneos, onde vários eventos podem ocorrer ao mesmo tempo contribuindo para alterar o estado do sistema;
- A necessidade de sincronização, pois o início de uma atividade requer o término de outra.

Uma série de sistemas de automação podem ser tratados por SEDs, entre os quais se incluem sistemas de manufatura, protocolos de comunicação, sistemas automatizados de tráfego, sistemas computacionais e de gerenciamento de dados [10].

No projeto de controle supervisório para a automação de sistemas de manufatura, diversas abordagens têm sido utilizadas, dentre as quais se incluem Linguagens Controláveis [9], Redes de Petri [11], Redes de Petri Controladas [12], Cadeias de Markov [13] e Teoria das Filas [14]. Diferentemente das outras abordagens que em sua maioria limitam-se à análise de sistemas, a modelagem por Linguagens Controláveis permite a síntese automática de controladores independente da experiência e inspiração do projetista [10].

A teoria de controle de SEDs proposta por [9] tem por base a modelagem da planta (sistema em malha aberta) e das especificações de controle, através de autômatos. Os autômatos podem ser ilustrados por diagramas de transição de estado, onde cada transição é determinada por um evento. Os de eventos que compõe os modelos individuais de cada subsistema físico são divididos em dois tipos de eventos [15]:

- eventos controláveis, cuja ocorrência pode ser desabilitada pela ação de controle, como por exemplo, o início de uma atividade ou a parada de uma esteira;
- eventos não-controláveis, cuja ocorrência não pode ser desabilitada pela ação de controle, como por exemplo, a ativação de um sensor.

RV E OS EVENTOS DISCRETOS

A implementação de eventos em RV é discutida em alguns trabalhos que tratam de simulação de eventos [16], simuladores de robôs [6], validação de modelos [17], sincronização de eventos [18]. Porém, em nenhuma delas é tratado o uso de eventos discretos. Na abordagem proposta neste artigo, a implementação em RV é feita tomando por base o resultado obtido a partir da teoria de controle supervisório, mas não há uma interface que permita a migração de SEDs para RV diretamente. Será mostrado como são mapeados os eventos discretos (controláveis ou não) a eventos de entrada e saída na tecnologia de RVNI usando VRML (que será apresentada na seção Eventos Discretos no VRML).

A CÉLULA DE MANUFATURA DIDÁTICA (CMD)

A concepção e a modelagem da CMD apresentada neste trabalho são descritas com maiores detalhes em trabalhos anteriores [19] e [20]. A CMD é composta pelos seguintes equipamentos (ver Figura 1):

- 2 Robôs Eshed Robotech Scorbot ER4PC;
- 1 Mesa giratória Intelitek (Rotary Table);
- 1 Esteira Intelitek (Conveyor ASSV);
- 1 Mesa de experimentos Intelitek (Experiment Table);
- 1 Estação de teste (Sensor fotoelétrico Sense tipo Dark on / Dark light).

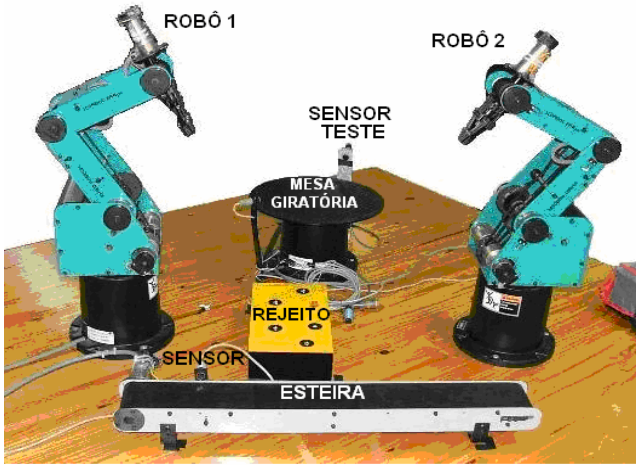

FIGURA 1
EQUIPAMENTOS DA CÉLULA

A seguir apresenta-se a descrição de uma seqüência de funcionamento da CMD. Na prática, outras seqüências são possíveis, porém fez-se uma simplificação para a elaboração deste trabalho.

Inicialmente uma peça é colocada sobre a esteira e um comando "início de funcionamento" liga a esteira. A peça é movimentada até o final da esteira, onde um sensor detecta a presença da mesma e gerando um sinal para desligar a esteira. O robô 1 é acionado e retira a peça da esteira, colocando-a na mesa giratória. A peça então é

movimentada pela mesa giratória até o próximo quadrante, onde um sensor de teste informa se a peça possui alguma não conformidade (representada por uma tarja preta na peça). A partir dessa informação, o robô 2 pode executar duas rotinas de movimentação. Caso a peça não possua a tarja, o robô 2 transporta a peça para uma área destinada a peças boas. Caso contrário, a peça é rejeitada e transportada para uma área de peças ruins. Após a conclusão das rotinas (uma ou outra) do robô 2, uma nova peça pode ser inserida na esteira.

EVENTOS DISCRETOS NO VRML

VRML

O VRML (*Virtual Reality Modeling Language)* é uma linguagem independente de plataforma que permite a criação de cenários 3D, por onde se pode passear, visualizar objetos sob diversos ângulos e ainda interagir com eles [21]. A linguagem foi concebida para descrever simulações interativas de múltiplos participantes, em mundos virtuais disponibilizados na Internet e ligados com a *World Wide Web*.

O VRML é um padrão (ISO 14772-1:1997) no qual os objetos podem ser representados através de geometrias primitivas, transformações hierárquicas, fontes de luz, pontos de visão, animações, mapeamentos de texturas, entre outros. O VRML permite a criação de mundos virtuais a partir de arquivos ".wrl" escritos em código ASCII. A fim de visualizar e interagir com ambientes e objetos virtuais, um *plugin* precisa ser instalado. O *plugin*, como extensão ao navegador *WWW*, será ativado toda vez que arquivos com a extensão wrl forem acessados [21].

Descrevendo Figuras

A construção do ambiente virtual é realizada com figuras (*shapes)* descritas pelos nós e por seus atributos. Desta maneira, a figura VRML tem uma forma ou geometria (*geometry*) que define a sua estrutura tridimensional. Além disto, esta figura tem sua aparência baseada no material com o qual é feita (*material*) e na textura da sua superfície (*texture*). Assim, *geometry* e *appearance* são os campos do nó *Shape*, conforme o exemplo do Quadro 1:

```
Shape {
    Appearance DEF Azul Appearance { # Campo da aparência da
figura
    material Material {
                    diffuseColor 0.0 0.0 1.0
                    }
        }
    geometry Cylinder {# Definição da sua forma 3D
        height 2.0
        radius 2.0
        }
}
```

QUADRO 1
UM EXEMPLO DE PROGRAMAÇÃO EM VRML

A linguagem VRML suporta vários tipos de primitivas geométricas para as figuras, de forma pré-definida, como caixas, cilindros, cones e esferas. A fim de modelar geometrias complexas utiliza-se o nó *IndexedFaceSet*. A partir de uma lista de vértices o *IndexedFaceSet* constrói faces que unidas representam um objeto em 3D [21]. Conforme descrito no trabalho de [6], na Figura 3 mostra-se um exemplo do *IndexedFaceSet* representando uma pirâmide. A regra para o sistema de coordenadas é o da mão direita, conforme visualizado na direção dos eixos. O caractere # antecede um comentário que é ignorado pelo visualizador de VRML. A propriedade *coord* contém um nó do tipo *Coordinate* que armazena um conjunto de coordenadas 3D representando os vértices do objeto. O primeiro vértice possui índice zero, o segundo possui índice um e assim sucessivamente. A propriedade *coordIndex* armazena índices com a qual é possível especificar as faces do polígono. A seqüência de vértices finalizada por –1 representa uma face. A Figura 3 demonstra o código onde se tem cinco faces representando as paredes e a base de uma pirâmide.

```
#VRML V2.0 utf8

Shape {
  geometry IndexedFaceSet  {
    coord Coordinate {
      point [
         1 0 -1, #vértice 0
        -1 0 -1, #vértice 1
        -1 0  1, #vértice 2
         1 0  1, #vértice 3
         0 2  0  #vértice 4
      ]
    }
    coordIndex [
      0 4 3 -1 #face direita
      1 4 0 -1 #face de trás
      2 4 1 -1 #face esquerda
      3 4 2 -1 #face frontal
      0 3 2 1 -1 #face inferior
    ]
  }
}
```

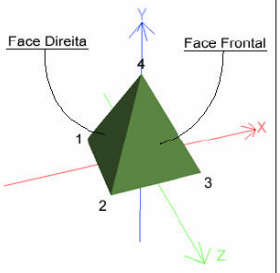

FIGURA 3
UTILIZAÇÃO DO NÓ *INDEXEDFACESET*

Eventos e Rotas

A linguagem VRML permite a criação de mundos dinâmicos onde eventos acontecem e são gerenciados. Para fazer isto, os arquivos VRML dispõem de instruções que descrevem como conectar nós para atuação conjunta ou condicionada. Por exemplo, abrir uma porta quando o usuário seleciona (clica) um certo objeto ou acender uma lâmpada nas mesmas condições.

Estes eventos são mecanismos de troca de mensagens entre os objetos e envolvem:

- um par de nós para atuar conjuntamente;
- uma rota de comunicação entre os dois nós envolvidos;

Uma vez a rota tendo sido definida entre os dois nós, o primeiro deles pode enviar mensagens ao segundo. Esta mensagem, chamada de *event* (evento) contém valores que forçarão o nó receptor a reagir. Existe inclusive a possibilidade de encadear vários nós criando um circuito de comunicação.

Cada tipo de nó apresenta acionadores de entrada e saída com os quais os eventos atuam. Por exemplo, o nó que cria uma luz (lâmpada) tem um acionador de entrada que pode ligá-la ou desligá-la (*on/off*). Acionando-o, através do envio de um evento, é possível ligar e desligar esta luz à distância.

É importante salientar que nós podem ter vários acionadores para interação entre nós do circuito. Por outro lado, estes acionadores são de entrada (input) e saída (output). Os acionadores de entrada são chamados de *eventIn* e os acionadores de saída são chamados de *eventOut*. Também é importante notar que estes acionadores têm um tipo de dado definido para receber ou enviar através dele.

Neste contexto, o circuito necessita descrever uma rota (*route*) de um acionador *eventOut* de um dado nó para um acionador *eventIn* de outro nó. O circuito permanecerá inativo até que um evento seja enviado entre os nós. A reação do segundo nó depende:

- do tipo de nó que está recebendo o evento;
- do acionador de entrada que foi ativado;
- do valor contido no evento;
- da atividade do nó no momento do recebimento do evento.

Os eventos são, geralmente, acionados por sensores. Sensores são nós que têm a capacidade de gerar eventos respondendo a ações do usuário (*ProximitySensor, VisibilitySensor, TouchSensor, CylinderSensor, PlaneSensor, SphereSensor* e o nó de agrupamento *Collision*) ou ao passar do tempo (*TimeSensor*).

Os nós sensores são nós que esperam um evento para realizar alguma ação em resposta. Um destes nós que se utilizou para implementar a CMD foi o *TouchSensor*. O nó *TouchSensor* define um sensor capaz de detectar ações do usuário sobre o objeto e produzir a saída correspondente a ação desejada. Através da seleção via *mouse*, este sensor possibilita que seja gerado um evento de saída (*eventOut*), chamado *touchTime.* Uma vez que um sensor tenha gerado um evento inicial, este evento é propagado através de quaisquer rotas para outros nós. Estes nós podem responder gerando eventos adicionais, e assim sucessivamente, através de um circuito de animação. As animações em VRML consistem em enviar eventos seguindo uma seqüência de passos.

O Quadro 2 mostra um exemplo de circuito de animação da CMD, onde são descritos os roteamentos para os nós *TouchSensor* e *TimeSensor*. O circuito de roteamento consiste em movimentar o objeto definido como Cubo pelo ambiente virtual. Para isso, é necessário definir uma trajetória através de pontos chaves (*keyValue),* e uma curva de aceleração que dá a forma em que o objeto percorre a trajetória, nos respectivos pontos chave (*key)*. Estes elementos compõem o nó *PositionInterpolator*. A quantidade de pontos da interpolação deve ser sincronizada com o tempo da animação, e isso é feito no nó *TimeSensor,* cujo tempo é dado em segundos. No exemplo, o intervalo de tempo escolhido é de 1 segundo.

O nó *TouchSensor* está relacionado a um objeto no mundo virtual, o qual ao ser "tocado", irá gerar um evento de saída chamado *touchTime.* Trata-se de um evento de saída que pode ser usado para determinar o início da contagem de tempo do objeto receptor a partir do momento que ocorre o toque. O objeto receptor será o nó *TimeSensor*, aqui chamado de *Clock*. O sensor *TimeSensor* gera eventos de tempo a partir do instante definido pelo campo *startTime*. Assim, um evento de toque pode iniciar a geração de eventos de tempo do *Clock*. Isso é descrito na primeira linha de roteamento do Quadro 2 (*ROUTE TOS.touchTime TO Clock.startTime* # Evento de toque => Início da animação). Na segunda linha de roteamento (*ROUTE Clock.fraction_changed TO PI.set_fraction* # Tempo => Cubo), é feita a sincronização do tempo com as posições do Cubo no mundo virtual. Assim, a cada fração de tempo do *Clock* é associada uma posição do Cubo. Na terceira linha de roteamento (*ROUTE PI.value_changed TO Cubo_inicial.translation* # Movimentação do Cubo), ocorre a movimentação do Cubo pelo mundo virtual, a partir da sua posição inicial. Dessa forma, durante o intervalo de tempo (1s) definido no nó *TimeSensor*, acontece a movientação do Cubo pelo ambiente virtual.

```
DEF Cubo Transform {...}          # Formato do Cubo
DEF PI PositionInterpolator {     # Interpolador de posições
    key [0, 0.25, 0.5, 0.75, 1] # Quantidade de pontos
    keyValue [ 15 32 -9.5,  15 24 -9.5,  15 6 -9.5,  15 2 -9.5, 15
                0 -9.5 ]          # Posições do Cubo

DEF TOS TouchSensor {}            # Sensor de toque
DEF Clock TimeSensor { cycleInterval 1 loop FALSE }

ROUTE TOS.touchTime TO Clock.startTime
ROUTE Clock.fraction_changed TO PI.set_fraction
ROUTE PI.value_changed TO Cubo_inicial.translation
```

QUADRO 2
CIRCUITO DE ANIMAÇÃO DA CMD.

A Célula de Manufatura Didática Virtual

Para a construção da CMD em VRML inicialmente foi feita a modelagem dos seus objetos. A CMD Virtual é composta por dois robôs, uma esteira de entrada de peças, uma mesa giratória, três botões (B1, B2 e B3) e dois locais para armazenamento de peças (ao lado do robô 2 e ao lado da mesa giratória).

Para dar início à seqüência de funcionamento na CMD Virtual, é necessário clicar com o *mouse* no botão B1, para que uma peça seja colocada sobre a esteira e a mesma seja ligada. A peça é movimentada até o final da esteira, onde o robô 1 é acionado e retira a peça da esteira, colocando-a na mesa giratória. A peça então é movimentada pela mesa giratória. Na CMD real, um sensor de teste informa se a peça possui alguma não conformidade (representada por uma tarja preta na peça). Na CMD virtual, o usuário é quem faz a escolha por peça boa ou ruim, clicando no botão B2 ou no botão B3. A partir dessa seleção, o robô 2 pode executar duas rotinas de movimentação. Caso a seleção seja o botão B2, o robô 2 transporta a peça para uma área destinada a peças boas (representada no ambiente por uma caixa ao lado do robô 2). Caso a seleção seja feita pelo botão B3, a peça é transportada para uma área de peças ruins (representada no ambiente por uma caixa ao lado da mesa giratória). Após a conclusão do movimento do robô 2, uma nova peça pode ser inserida na esteira clicando-se novamente no botão B1. Na Figura 4 é apresentado o ambiente virtual da CMD.

FIGURA 4
A CMD VIRTUAL

CONSIDERAÇÕES FINAIS

Foram vistos neste trabalho conceitos sobre Sistemas a Eventos Discretos e Realidade Virtual, a forma como são relacionados os eventos no mundo real (controláveis e não controláveis) e no mundo virtual (eventos de saída e eventos de entrada) e como estes eventos são implementados na prática. Foi detalhada a criação em VRML de uma Célula de Manufatura Didática, tomando-se por base uma célula de manufatura real, existente em um laboratório de ensino. Limitou-se o uso da RV à visualização e simulação de eventos através da interação do usuário utilizando o *mouse*. Uma possível extensão deste trabalho é o desenvolvimento de uma interface que permita a comunicação da célula virtual com a célula real, viabilizando a monitoração e interação com a célula de manufatura didática via internet.

REFERÊNCIAS

[1] Kirner, C. e TORI, R. Editores (2004). *Realidade Virtual*: Conceitos e Tendências. São Paulo, Livro do Pré Simpósio SVR. pp 3-8, ISBN 85-904873-1-8.

[2] Latta, J. N. & Oberg, D. J. (1994). A conceptual virtual reality model. IEEE Computer Graphics & Applications, pp. 23-29.

[3] Pimentel, K. e Teixeira, K. (1995).Virtual Reality – through the new looking glass. 2.ed. New York, McGraw-Hill.

[4] Vince, John. (1998). Essential Virtual Reality Fast. Berlin, Alemanha, Ed. Springer, 174 pgs, ISBN: 1-85233-012-0.

[5] Hand, C (1994). Other faces of virtual reality, First International Conference MHVR'94 - Lecture Notes in Computer Science n.1077, pp. 107-116, Ed. Springer, Moscow, Russia, September.

[6] Hounsell, M. S. e REDEL, R. (2004). Implementação de Simuladores de Robôs com o Uso da Tecnologia de Realidade Virtual. IV Congresso Brasileiro de Computação – CBComp 2004.

[7] Jacobson, L. (1994). Realidade virtual em casa. Rio de Janeiro, Berkeley.

[8] Montgomery, E. (2004). Introdução dos Sistemas a Eventos Discretos e à Teoria de Controle Supervisório. Rio de Janeiro. Ed. Alta Books. 122 pgs, ISBN: 857608065-6.

[9] Ramadge, P. J. e Wonhan, W. M. (1989). The Control of Discrete Event Systems, Proceedings of IEEE, 77(1), pp. 81-98.

[10] Queiroz, M.H. et al. (2001). Síntese modular do controle supervisório em diagrama escada para uma célula de manufatura. Anais do V Simpósio Brasileiro de Automação Inteligente, Vol. 5, p. 1072.

[11] Murata, T. (1989). Petri Nets: Properties, Analysis and Aplications. Proceedings of the IEEE, 77(4):541-580.

[12] Krogh, B. H. e Holloway, L. E. (1991) Synthesis of Feedback Control Logic for Discrete Manufacturing Systems. Automatica, 27(4):641-651.

[13] Çinclair, E. (1975). Introduction to Stochastic Processes. Prentice-Hall, Inc., Englewood Cliffs, N.J., USA.

[14] Kleinrock, L. (1975). Queueing Systems, Volume I: Theory. John Wiley & Sons, Canada.

[15] Cury, J. E. R. (2001). Teria de Controle Supervisório de Sistemas a Eventos Discretos. V Simpósio Brasileiro de Automação Inteligente (Minicurso). Canela, RS.

[16] GABCAN, L. et al. (2002). Utilização de Técnicas de Realidade Virtual na Visualização de Simulação de Atendimento em Hospital. I Workshop de Realidade Virtual e Visualização Científica do Laboratório de Métodos Computacionais em Engenharia. Rio de Janeiro, 2002.

[17] Perkusich, A. e Silva, L. D. (2003). Uso de Realidade Virtual para Validação de Modelos de Sistemas Flexíveis de Manufatura. VI Simpósio Brasileiro de Automação Inteligente. Bauru, SP.

[18] Souza, J.L.R. e Kniess, J. (2004). Apresentação e Implementação de um modelo para especificar Sincronização em Realidade Virtual. IV Congresso Brasileiro de Computação – CBComp 2004.

[19] Curzel, J. L. e Leal, A. B. (2006). Implementação de Controle Supervisório em Linguagem Ladder para uma Célula Flexível de Manufatura Didática. XVI Congresso Brasileiro de Automática. pp 2700-2705. Salvador, BA.

[20] Curzel et al. (2006). Concepção de uma Célula de Manufatura Didática para o Ensino de Engenharia. XXXIV Congresso Brasileiro de Ensino de Engenharia. pp 1916-1926. Passo Fundo, RS. ISBN 85-7515-371-4

[21] Carrey, R. e Bell, G. (1997). The annotated VRML 2.0 reference manual. Mass.: Addison-Wesley Developers Press, ISBN 0-201-41974-2.