MODELING AND SIMULATION OF FMS DYNAMICS BY USING VRML

N. Smolić-Ročak, S. Bogdan, Z. Kovačić
T. Reichenbach, B. Birgmajer

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, CROATIA
http://flrcg.rasip.fer.hr

Abstract: This paper presents advantages of using virtual reality FMS models and corresponding dynamic simulators for design, analysis, dynamic simulation and visualization of complex flexible manufacturing systems. Developed software has been used for creating a virtual model of the laboratory FMS setup. It is shown how phenomena like operation conflicts and deadlocks can be observed and supervised remotely through a client-server Internet connection.

Keywords: flexible manufacturing systems, modeling, simulation, virtual reality

1. INTRODUCTION

Flexible manufacturing systems (FMS) can be viewed as complex structures assembled of elements such as robots, machine tools, rotary tables, belt conveyers etc., that are connected and supervised through a local area network access. Design of FMS can be a demanding job and for this purpose graphically oriented tools for design of FMS have been developed. To get an insight into the dynamic performance of FMS, the methods for modeling and control based on the discrete event systems theory are applied. Good examples are the programs Onika (Gertz and Khosla, 1994), Robotica (Nethery and Spong, 1994), OpenRob (Ge, *et al.*, 2000) and the Robotics Toolbox for Matlab (Corke, 1999), which allow graphical design of systems for control of robotized plants and which integrate already existing software modules for control of robot manipulators.

Virtual modeling of complex physical systems has brought a new quality in investigation of FMS phenomena. Allowing clear visualization of all problems arising during FMS operation, virtual reality modeling in conjunction with dynamic characteristics of the modeled objects and web-related technologies has traced a completely new route to analysis and design of flexible manufacturing processes (Jacobs, *et al.*, 1996; Hirukawa and Hara, 2000).

The aim of this paper is to show how effective can be analysis of FMS dynamic behavior with the usage of virtual models and accompanying dynamic models. By using a developed software, based on JAVA and C++, a virtual model of the previously built laboratory model of the FMS has been made to illustrate the occurrence of critical phenomena such as the operations conflict and deadlock, which endanger regular FMS operation. Being clearly visualized, these irregularities can be prevented by selecting an appropriate dispatching policy.

2. FLEXMAN CONCEPT

FlexMan is a tool for computer-integrated design and simulation of FMS (Kovačić, *et al.*, 2001). Its concept is directed towards concretization of a very popular term “future factory” which assumes a high level of integration of the manufacturing tasks (strategies and production planning) on one hand and the machine infrastructure (robots, transporters, machine tools) on the other hand. The aim of FlexMan is to provide an environment for investigation of various shop floor configurations and job scheduling strategies with one goal - to reach a necessary level of manufacturing flexibility.

The concept of FlexMan has been based on five main software components (Fig. 1): the Scene Builder, the trajectory planner tool LEONARDO, the Database, the Visualization Client and the FMS Controller. Although some components can operate independently, the interconnection and communication between various parts under supervision of the FMS controller is what makes the whole concept feasible.

The software is based on the client - server architecture (actually, there is more than one server and some servers can also act as clients) with thin "final" client (that serves for visualization of a virtual FMS). The servers are implemented in C++ for the win32 platform, while the clients can be any Internet browsers that support the JAVA technology and have incorporated the VRML.

Although the data are transferred with different protocols, they are all stored in XML format. The advantages of such an implementation are easy software maintenance, scalability, adaptability to customer needs (easy reconfiguration of the virtual scene), the usage of standard protocols, no need for "extra" client software - standard HTML browsers with VRML viewer plug-ins and Java support can be used, and finally, the important thing is that the client can be run on various hardware platforms.

The ***Scene Builder*** is a component written in Java, which serves for graphically based setting of virtual FMS configurations. Elements such as robots, conveyers, pistons, machines, tools etc., whose virtual models are used in the scene, are described in the Database. The ***Database*** consists of the records of tools, robots, parts, trajectories, scenes and saved statistical information.

LEONARDO is an integrated tool for the complex objects off-line trajectory planning, that is primarily applicable to the single robot work cells.

The ***Visualization client*** is responsible for accurate visualization of a simulated virtual FMS. The commands received from the FMS Controller are used to move the parts and operate the tools. The Visualization Client is built completely in JAVA. Control of the VRML scene, which is a representation of the FMS model, is achieved via the External Authoring Interface (EAI).

The ***FMS Controller*** serves as a supervision and communication center and on top of that as a virtual FMS simulation executor. The FMS planner uses a timed matrix-based model (Bogdan, *et al.*, 1999) to determine which operation is taking place and what is the duration of an operation. Once the FMS scene is defined and selection of the FMS control strategy has been made, the FMS planner does not require any further interaction with the user. All data necessary for simulation and visualization are calculated based on the parameters in the Database. Different FMS control strategies can be applied and their impact on the behavior of the virtual FMS can be investigated.

3. A TIMED MATRIX-BASED MODEL OF FMS

FlexMan utilizes a timed matrix-based algebra (Bogdan, *et al.*, 1999) to simulate FMS dynamics. In order to create a timed matrix-based model of an FMS it is required first to define a sequence of manufacturing operations. The sequence of operations can be generated on the basis of a part path, and then easily transformed into the matrices $\mathbf{S_r}$ - resource release matrix, $\mathbf{S_v}$ - job start matrix, $\mathbf{S_y}$ - output matrix, $\mathbf{F_r}$ - resource requirements matrix, $\mathbf{F_v}$ - job sequencing matrix and $\mathbf{F_u}$ – input matrix, that are explained in detail in (Lewis, *et al.*, 1998). These matrices describe logical connections among the resources and operations, or in other words, they represent the set of IF-THEN rules.

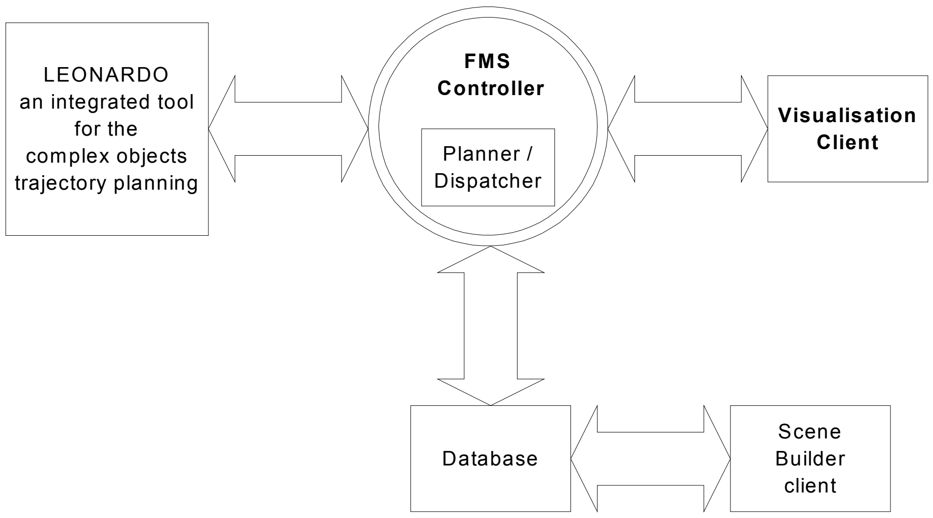


Fig. 1. The concept of FlexMan.

However, every operation takes a specific time to get completed, so it is necessary to define durations of all operations. From the time intervals specified for each operation corresponding diagonal matrices $\mathbf{D_r}$ and $\mathbf{D_v}$ containing operation delays are created.

According to the timed matrix-based FMS model the number of parts waiting to proceed through a flow line after completion of a corresponding job (a component of job vector **v**) can be calculated by using the following recursive equation:

$$\left(1-z^{-1}\right)v(z)=\left(T_v(z)-F_v^T\right)x(z) \qquad (1)$$

where:

$T_v(z)=D_v^*(z)S_v$ - task duration matrix.

In the same way the number of available resources (components of resource vector **r**) can be determined as:

$$\left(1-z^{-1}\right)r(z)=\left(T_r(z)-F_r^T\right)x(z) \qquad (2)$$

where:

$T_r(z)=\Phi D_r^*(z)S_r^*$ - resource release duration matrix,

Φ - release transformation matrix, see (Bogdan, *et al.*, 1999).

Components of the logical state vector **x**, which define operations to be started and resources to be released (referring to (1) and (2)), depend on a current status of operations and resources. They are calculated from the set of IF-THEN rules, which can be represented as a recursive equation of the form

$$x(z)=\sigma[y(z)] \qquad (3)$$

where:

$$y(z)=F\sigma\left[z^{-1}m(z)\right]-\left(w-\begin{bmatrix}1\\ \dots\\ 1\end{bmatrix}_{n\times 1}\right)$$

$$w=F\begin{bmatrix}1\\ \dots\\ 1\end{bmatrix}_{n\times 1},\quad m(z)=\begin{bmatrix}v(z)\\ r(z)\end{bmatrix},\ \sigma(a)=\begin{cases}1, & a>0\\ 0, & a\le 0\end{cases},$$

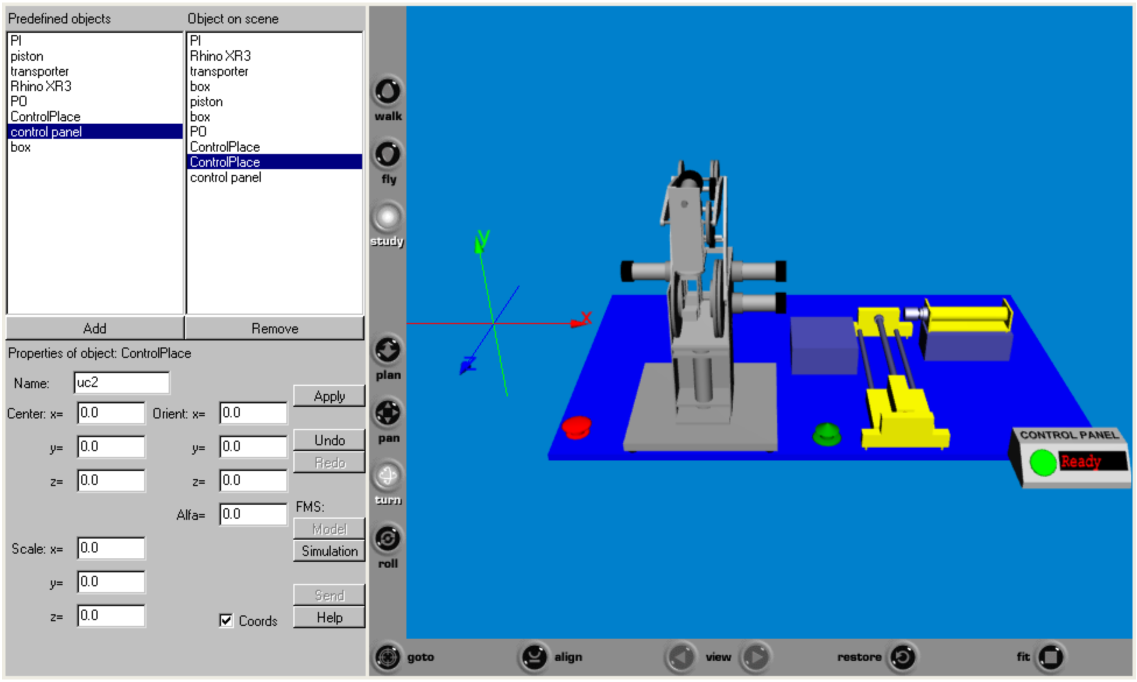
$$F=\left[F_v \;\vdots\; F_r\right]$$

To resolve a possible occurrence of operations conflict due to the existence of shared resources in the FMS, controllers must be used. The FMS controller is integrated within a timed matrix-based model through the user-defined control matrices $\mathbf{S_d}$ and $\mathbf{F_d}$. The structures of these two matrices are determined by the structure of the controlled FMS and the desired dispatching strategy (Gurel, *et al.*, 2000)

4. VIRTUAL MODELING OF FMS

As a part of FlexMan, the Scene Builder is an easy-to-use Internet browser-based tool for virtual scene customization and FMS properties definition. To create a virtual model of an FMS, a user must open a specified URL address by using one of available browsers (e.g. Netscape, Internet Explorer).

A virtual scene (Fig. 2.) is modeled by using objects from the Database (Predefined objects list) whose characteristics are defined by the corresponding prototypes. In this way, a build-up of more complex layouts becomes very easy and straightforward.



Predefined objects: PI, piston, transporter, Rhino XR3, PO, ControlPlace, control panel, box

Object on scene: PI, Rhino XR3, transporter, box, piston, box, PO, ControlPlace, ControlPlace, control panel

Properties of object: ControlPlace

Name: uc2

Center: x= 0.0, y= 0.0, z= 0.0

Orient: x= 0.0, y= 0.0, z= 0.0, Alfa= 0.0

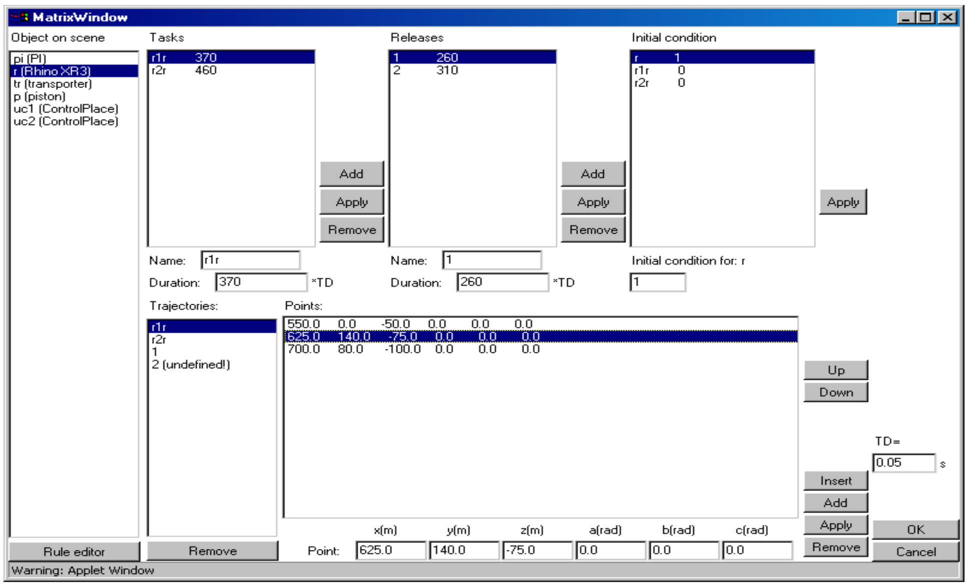
Scale: x= 0.0, y= 0.0, z= 0.0

[✓] Coords

Fig. 2. A virtual scene of an FMS created with a Scene Builder tool.

Following the VRML specifications (X3D Consortium, 2001), the prototype created for FMS elements defines their position and orientation and a scaling factor that determines their respective size. Depending on the structure of each element and the number of its movable parts (e.g. the number of axes), the corresponding prototype defines interface for external manipulation of the object through definition of so called eventIn input variables.

result in a clear picture of what is going on in the FMS during the manufacturing process. Providing that resources and operations they perform are defined, to make a virtual model feasible for simulation, the user has to define FMS operation sequencing. This can be done by a Rule Editor frame (Fig. 4). From the previously defined objects and their tasks the user builds a set of rules that describes sequencing of operations in the FMS:



Object on scene:
pi (PI)
r (Rhino XR3)
tr (transporter)
p (piston)
uc1 (ControlPlace)
uc2 (ControlPlace)

Tasks	
r1r	370
r2r	460

Name: r1r
Duration: 370 *TD

Releases	
1	260
2	310

Name: 1
Duration: 260 *TD

Initial condition	
r	1
r1r	0
r2r	0

Initial condition for: r
1

Trajectories:
r1r
r2r
1
2 (undefined!)

Points:

```
550.0   0.0    -50.0   0.0    0.0    0.0
625.0   140.0  -75.0   0.0    0.0    0.0
700.0   80.0   -100.0  0.0    0.0    0.0
```

TD= 0.05 s

	x(m)	y(m)	z(m)	a(rad)	b(rad)	c(rad)
Point:	625.0	140.0	-75.0	0.0	0.0	0.0

Warning: Applet Window

Fig. 3. Object task definition frame (Rhino XR3).

Once an object is placed on the virtual scene, its properties (name, position, orientation etc.) can be defined by using the corresponding fields. Some objects on the scene, such as resources, have a physical meaning while the others, such as control place, serve only as logical entities for the purpose of FMS control. It is worth to mention that the object called “CONTROL PANEL” (in the right bottom corner of Fig. 2) displays a current status of the FMS notifying the user if for example, a conflict or a deadlock have occurred.

Having a structure of the FMS described by the virtual model, the user opens a frame for definition of tasks performed by the objects on the scene. Depending on a selected type of an object, different frames are opened (Fig. 3). For example, as shown in Fig. 3, the object named “r” represents a virtual model of the educational robot Rhino XR-3 (see Fig. 2), which has two tasks, “r1r” and “r2r”. The frame allows the user to define all data that determine these tasks (trajectories, resource release durations, durations of tasks etc.). On the other hand, the frame for the Control place enables only definition of object’s release time and initial condition.

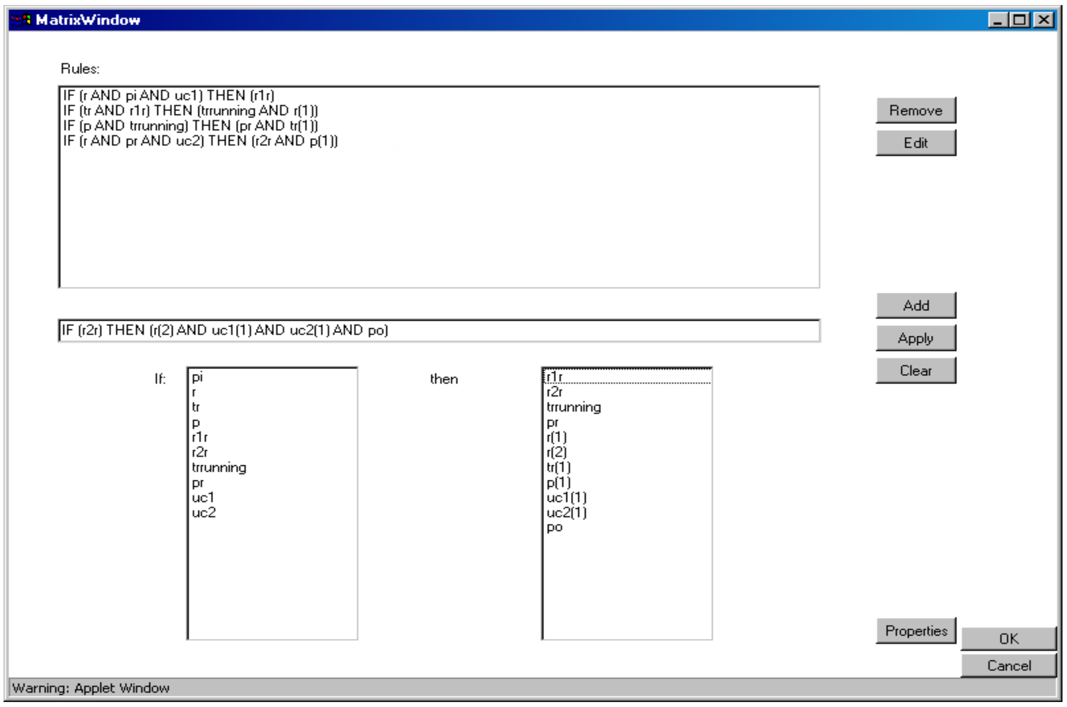
In order to visualize FMS operations in the virtual environment as they were real, the algorithm described by equations (1)-(3) must be active and its input and output must be closely connected to the elements of the virtual model. This will eventually

IF operation_1 is finished AND resource_1 is ready THEN start operation_2 AND release resource_2. Based on these rules and objects properties the FMS model matrices required for simulation are calculated automatically.

The elements of matrix $\mathbf{F_r}$, which relates resources and IF part of rules, attain value 1 if the corresponding resource (column) participates in a rule (row). Otherwise the corresponding element becomes 0. In the same way, matrix $\mathbf{F_v}$ gets 1s in the places where the corresponding task (column) participates in the IF part of rule (row). THEN parts of the rules are related to the resources and operations through matrices $\mathbf{S_r}$ and $\mathbf{S_v}$. If the resource (row) takes part in a rule (column), then the corresponding element of matrix $\mathbf{S_r}$ gets value 1. The same is effective for matrix $\mathbf{S_v}$ in which rows represent tasks while columns represent rules.

Trajectories for resources with only one degree of freedom are generated on-line by the FMS planner, while the program LEONARDO generates trajectories for more complex resources off-line.

The connection between the Scene Builder and the FlexMan is accomplished through XML formatted data. When the complete FMS is properly initialized, simulation can start. During simulation the FMS Controller sends coordinates and orientation of a tool tip for each resource and part to



```
Rules:
IF (r AND pi AND uc1) THEN (r1r)
IF (tr AND r1r) THEN (trrunning AND r(1))
IF (p AND trrunning) THEN (pr AND tr(1))
IF (r AND pr AND uc2) THEN (r2r AND p(1))

IF (r2r) THEN (r(2) AND uc1(1) AND uc2(1) AND po)

If:          then
pi           r1r
r            r2r
tr           trrunning
p            pr
r1r          r(1)
r2r          r(2)
trrunning    tr(1)
pr           p(1)
uc1          uc1(1)
uc2          uc2(1)
             po

Warning: Applet Window
```

Fig. 4. The Rule Editor frame.

the Visualization Client, and correspondingly, the Visualization Client moves resources and parts to defined positions. FMS Controller sends data to Visualization Client only when there is a change in the state of object. The data are sent in the XML format via socket communication. Data are accepted by the Visualization Client, which parses the XML document and puts parts and resources in positions given in the XML document.

5. AN EXAMPLE

An example of FMS, which contains a Rhino XR-3 robot, three transporters and three pistons are shown in Fig. 5. (a dotted line represents a part path).

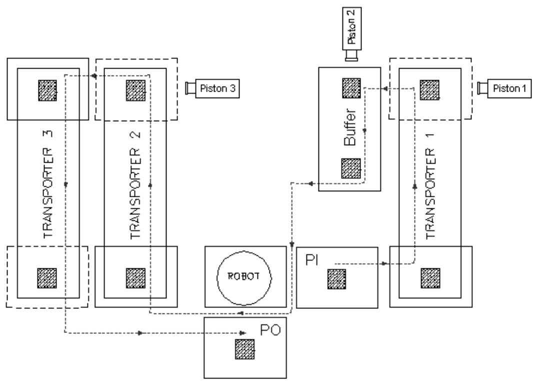
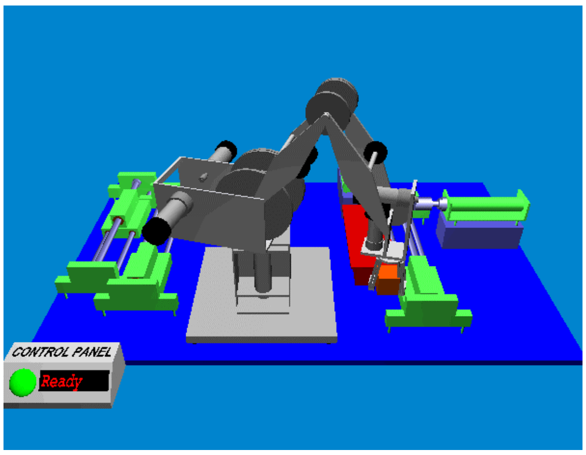


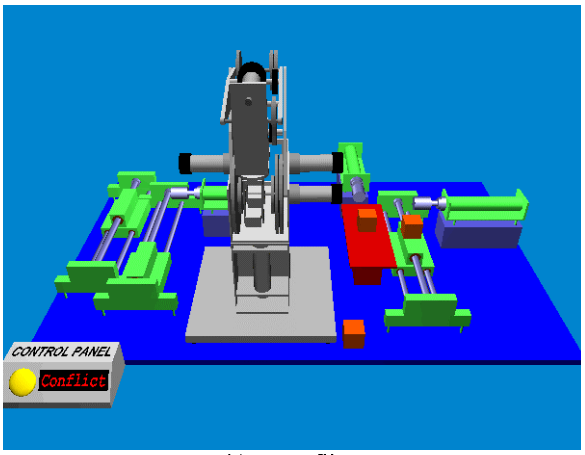
Fig. 5. The machine shop of experimental FMS.

A virtual model of this FMS in various states of operation is presented in Figs. 6a-c. Fig. 6a. illustrates starting operations where the robot takes the first part from the input feeder, carries it and puts it on the transporter T_1, which further carries it in front of the piston P_1. A simulator, using the matrix-based model, provides simultaneous execution of more than one operation.

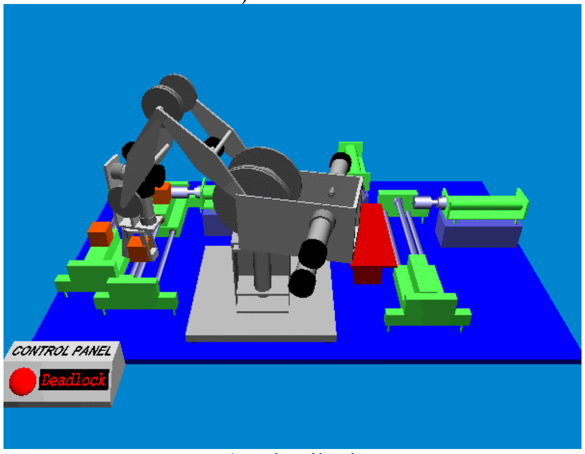
Fig. 6b. illustrates the state of conflict as the robot must decide which of the two possible jobs should do first – to take the part from the buffer and put it on the transporter T_2 or to take another part from the input place and put it on the transporter T_1. Such a conflict is easily resolved by an adequate controller action that will enable one and disable another possible operation. In the studied case the FMS controller successfully resolves the situation by deciding that the robot takes another (the third) part and carries it to the transporter T_1.



a) the beginning of operation



b) a conflict



c) a deadlock

Fig. 6. A virtual model of an experimental FMS.

Fig. 6c. shows a deadlock. The robot must remove the part from T_3 while holding a part that cannot be put on already occupied transporter T_2. This stopped parts to proceed through the system. Obviously, the user must use a Scene Builder to change a dispatching policy by editing and changing the rules.

6. CONCLUSIONS

Virtual modeling of complex physical systems has given a new dimension to the FMS design. The usage of virtual FMS models enables easy changes of FMS configurations and allows feasibility studies. All phenomena can be clearly visualized in the virtual world and possible conceptual mistakes can be eliminated before building the real system.

By providing a powerful 3D visualization, virtual models enable deeper insight and understanding of an FMS dynamic behavior, thus becoming very useful for education and personnel training. The graphical user interface supporting 3D graphics and animation provides facilities for off-line elaboration and validation of various virtual laboratory and virtual factory concepts. For this purpose the program FlexMan has been developed and presented with an additional feature of supporting remote control via Internet through the client-server communication.

The usage of virtual models provides conditions for realistic visualization of conflict situations and thus helps to explore all possible situations and implement other dispatching strategies, which will ensure that the system is deadlock free.

7. ACKNOWLEDGEMENTS

The work described in this paper was performed within the framework of the projects 00-2 "Design of FMS by Using Virtual Models and Internet" and 036042 "Integrated Control of Robotized Systems" supported by grants from the Ministry of Science and Technology, Republic of Croatia.

REFERENCES

Bogdan, S; F. Lewis, Z. Kovačić, A. Gurel, (1999). "New Matrix Formulation for Supervisory Controller Design in Practical Flexible Manufacturing System", *The 1999 IEEE ISIC*, Boston, pp. 144-149.

Corke P., (1999). Robotic Toolbox for Matlab, CSIRO Manufacturing Science and Technology

Ge S.S., Lee T.H., Gu D.L. and Woon L.C., (2000) "A One Stop Solution in Robotic Control System Design", *IEEE Robotics and Automation Mag.*, vol. 7, no. 3, pp.42-54.

Gertz M. W. and Khosla P. K., (1994). "Onika: A multilevel Human-Machine Interface for Real-Time Sensor-Based Robotics Systems", *in Proc. of SPACE 94,* Albuquerque.

Gurel A.; Bogdan S.; Lewis F.L.; Huff B. (2000); Matrix Approach To Deadlock-Free Dispatching In Multi-Class Finite Buffer Flowlines, *The IEEE Transaction on Automatic Control,* Vol. 45, No. 11, pp. 2086-2090.

Hirukawa H. and Hara I., (2000) "Web-Top Robotics:
Using the World Wide Web as a Platform for Building Robotic Systems", *The IEEE R&A Magazine*, vol. 7, no. 2, pp. 40-45.

Jacobs K., Lemay L, Murdock K. and Couch J., (1996) "Laura Lemay's Web Workshop: 3D Graphics and VRML 2", Sams Publishing.

Kovačić, Z, S. Bogdan, T. Reichenbach, N. Smolić-Ročak, B. Birgmajer, (2001). "FlexMan – A Computer-integrated Tool for Design and Simulation of Flexible Manufacturing Systems", *CD-proc. of The 9th Mediterranean Conference on Control and Automation*, Dubrovnik.

Lewis, F., A. Gürel, S. Bogdan, A. Doganalp, O. C. Pastravanu, (1998) "Analysis of Deadlock and Circular Waits Using a Matrix Model for Flexible Manufacturing System", *Automatica*, vol. 34, no. 9, pp. 1083-1100.

Nethery J., and Spong M.W. (1994), "Robotica: A mathematica package for robot analysis", *IEEE Robotics and Automation Mag.*, vol. 1, no. 1, pp.13-20.

X3D Consortium (2001). VRML 2.0 modeling language specification.