PETRI NETS AND GRAPHIC SIMULATION FOR THE VALIDATION OF COLLABORATIVE ROBOTIC CELLS IN AIRCRAFT INDUSTRY

Adriano José Cunha de Aguiar, ajca@ita.br
Emília Villani, evillani@ita.br
Instituto Tecnológico de Aeronáutica – ITA
Divisão de Engenharia Mecânica-Aeronáutica
Pça. Mal. Eduardo Gomes, 50 - Vila das Acácias
CEP: 12228-900 – São José dos Campos/SP, Brazil

Fabrício Junqueira, fabri@usp.br
Escola Politécnica, University of São Paulo, São Paulo, Brazil.

Abstract. *Today, the design of new robotic cells requires the use of graphic simulation tools that make possible the verification and definition of a number of key design points before having the physical devices (robots) and the workcell. Examples are to test the robot reachability, visualize the trajectory, detect collision points, support the definition of layout, determine production times, etc. In this context this paper addresses the issue of designing supervisory control systems for flexible collaborative robotic cells in aircraft industry. For this purpose a mixed approach based on Petri net and graphic simulation is proposed. Petri net is used to model the supervisory system that coordinates the activities of robots. Graphic simulation is used to illustrate the behavior of the robots in a 3D environment. The flexibility of the application requires that the robots work as 'slaves' of the supervisory system. Its trajectory is not previously programmed in the robot controller but is defined, in real time, by the supervisory system. As a consequence, in order to validate the collaborative robotic cell, both simulation tools must be integrated. The motivation of this work is the design of a workcell for the aircraft industry. The workcell automates the processes of drilling and clipping fuselage parts. The workcell is composed of two robots which must work in cooperation.*

Keywords*: Petri nets; Graphic simulation; Collaborative robotic; Validation; Aircraft industry; Robots; Supervisory system*

1. INTRODUCTION

The demand for manufacturing cells composed of multiple robots is becoming a frequent situation in industry. In these cases, the operations performed by one robot depend on the other robots. The integration and synchronism of the robots are a critical factor, as well as the integration of the robots with other machines of the manufacturing cell. According to Chaimowicz (2002), some types of tasks, called "strongly coupled tasks," can only be performed by the multiple robots operating in cooperation. In this case, the robots must be coordinated. When a robot executes a task, it must receive and provide information about its state to the other robots, in order to preserve its integrity and, also, the integrity of the entire system. Furthermore, the occurrence of failures in at least one of the robots can harm/hinder the fulfillment of the task of all.

The exchange of information and the coordination of the robots can be made by a supervisory system. In this case, the programming of the supervisory system and the programming of each robot must include a validation step that integrates the behavior of the robots and supervisory system in a common validation environment.

In this context, this paper addresses the problem of designing flexible robotic cells composed for multiple robots with a supervisory system responsible for integration and coordination of the robots.

The flexibility of the robotic cells is characterized by the following conditions:

- The definition of the robot trajectory is performed in real time by a system external to the robot controller. This is the case of applications that requires accuracy higher than that provided by the robot. An external measurement system then requires the correction of the robot position and orientation.
- The cooperation between multiple robots results in an undefined trajectory, requiring the interaction with the supervisory system to coordinate the activities of the robots.
- The application requires the processing of exceptions and failure treatment in a more elaborated way than simply suspending the operation of the robot.

This paper proposes a new approach to support the supervisory system design and validation based on the combined use of Petri nets and graphical simulation of robots. The Petri net is used for modeling and simulating the control logic of the supervisory system, while the graphical simulation of robot display the resulting behavior of the robots in a three dimensional environment.

This work is motivated by a cooperative project between ITA and the Brazilian aircraft industry. This project aims at the automation of aircraft structural assembly using multiple and flexible robotic cells.

This article is organized as follows. Sections 2 and 3 introduce the two techniques used in the proposed approach: graphical simulation of robots and Colored Petri Nets. Section 4 presents the proposed approach and illustrates its application through a case study of the aircraft industry. Finally Section 5 presents some conclusions and discusses future work.

2. DIGITAL MANUFACTURING AND GRAPHICAL SIMULATION OF ROBOTS

Today, the manufacturing organizations around the world, not only in the aeronautics sector but also in other sectors such as automotive, compose a hyper-competitive environment that forces a changing of paradigm. Products are becoming more complex and dynamic. Along with reduced cycles of innovation, product lifecycles and times-to-market are being progressively reduced (Souza *et al.*, 2002).

Companies are seeking new ways to achieve competitive advantage and bring new products to market faster and at a lower cost. One of the new paradigms that emerges to support this scenario is Digital Manufacturing. According to Banerjee and Zetu (2001), Digital Manufacturing can be defined as the modeling of systems and components with the effective use of computers, audiovisual devices and sensors to simulate or design alternatives for a manufacturing environment. Its purpose is to predict potential problems and inefficiencies in its functionalities or manufacture, before it is actually manufactured.

Digital manufacturing is the ability to describe every aspect of the design-to-manufacture process digitally—using tools that include digital design, CAD, CAM, analysis software, simulation, and so on. Digital Manufacturing should address the development, simulation and manufacturing of a product virtually on a computer before they are performed in the real world, regardless of the degree of complexity of the shape and the structure of the product (Souza *et al.*, 2002). The basic idea behind it is to move bits instead of moving atoms.

According to Souza *et al.* (2002), the use of Digital Manufacturing is being incorporated in some companies, especially in the aerospace and automotive sector. Some companies that already use the Digital Manufacturing are: Boeing, which developed the 777 aircraft in a full digital environment before actually building it; DaimlerChrysler, which produced three vehicles using Digital Manufacturing; and John Deere, which also has used this new environment in the development of its products (Wave, 2002). In the national scenario, many companies such as Volkswagen Brasil and ZF do Brasil are already using Digital Manufacturing with the purpose of optimizing processes, reducing development time and investment, improving quality, and managing knowledge.

Among the various resources available in Digital Manufacturing environment, the graphical simulation of robots stands out (Zimmermann, 2007). It is used for off-line programming, accessibility analysis of the tool, detection of interferences, among other. The graphical simulation of robots, especially of industrial manipulators, is increasingly used in manufacturing systems. The ease with which it incorporates the off-line programming makes the simulation a powerful tool for the planning of a new work cell or aiding in the programming of the current robots (Silva, 2004).

The graphical simulation of robot aims to visualize and check the performance of robots in a manufacturing cell. It can be used to determine some features, such as the robot reachability and envelope (Stobart and Dailly, 1985) (Aguiar *et al.*, 2007a).

Other benefits brought by the use of graphical simulators are (Silva, 2004), (Aguiar *et al.*, 2007b):

- Reduction in production times. With the help of simulation, it is possible to determine the time of each operation, detect bottlenecks, and seek for best trajectory solutions.
- Verification of accessibility. Graphical simulators come with libraries composed of a large range of commercial robot models. Therefore, it is possible to test the access of different robots and compare solutions without having the robots or building the work cells.
- Programming reuse and flexibility. It is possible to modify and reuse programmed operations. Regular and symmetric parts can be programmed using the mirror function.

A computational tool for graphical robot simulation must include from models of different robots and structures to inverse kinematics algorithms. The inverse kinematics supports the programming of the tool position or trajectory, in Cartesian space, without the need of defining the joints angles. It leaves to the simulator the responsibility of calculating the joint angles, given the position of the tool (Aguiar *et al.*, 2007b). Despite the resources provided by graphical simulators, the planning of trajectories for a robotic manipulator is still a complex task that involves various aspects such as modeling barriers, manipulating sensor data, searching for collision-free trajectories and avoiding settings that cause singularities in the robot (Leng and Chen, 1997).

Currently, a large number of commercial tools support graphical simulation and off-line programming of robots. Usually, these products are developed by companies that also produced CAD tools and/or companies that design and commercialize robots (Silva, 1996). Examples are ROBCAD, RobotStudio, Easy-Rob, KUKA.Sim Pro and Grasp10.

3. COLORED PETRI NETS

The second technique used for verification in the proposed approach is Petri Nets. Petri Nets (PNs) are a graphical and mathematical modeling technique originally developed by C.A. Petri in the early 1960s to characterize concurrent operations in computer systems. The greatest appeal of PNs is their conceptual simplicity (Moore and Brennan, 1996).

It is also good to describe static and dynamic system characteristic, and system uncertainty. The structure of PN models can be exploited to develop efficient algorithms for system control (Qiao et al., 2002).

PNs have been extended to capture many important aspects of systems, which includes attributes, timing relationships, and stochastic events (Moore and Brennan, 1996). Colored Petri Nets (CPNs) extend the classical Petri Nets with colors (to model data), time (to model durations), and hierarchy (to structure large models) (Mulyar and Van der Aalst, 2005). In real-world systems, we often find many parts that are similar. These parts must be represented by disjoint and identical sub-nets in PNs. This means that the net becomes largely and it becomes difficult to see the similarities between the individual sub-nets. CPNs provide a more compact representation where individual sub-nets are replaced by one sub-net with different kind of tokens, each token having a color and representing a different sub-net in the equivalent PN (Elkoutbi and Keller, 1998) (Qiao et al., 2002). CPNs are very appealing and appear uncomplicated because of their simple graphical representation (Arjona and Bueno, 2003). Like in classical Petri Nets, CPNs use three basic concepts: transition, place, and token (Mulyar and Van der Aalst, 2005) (Jensen, 1997a). CPN is a graphically oriented modeling language capable of expressing concurrency, non-determinism, and system concepts at different levels of abstraction. CPNs combine PN and programming languages within the same mathematical framework. Petri Nets are used to model concurrency, synchronization, and resource sharing and allocation, whereas a functional programming language is used to model data manipulation and to create compact and parameterized models (Zhang et al., 2001).

Places are used to store entities. Color sets associated with the places indicate the different types of entities that can be stored in the places. Entities are represented by color instances (copies) of the colors associated with the places and are referred to as colors or tokens. Tokens stored in the places, referred to as the marking of the CPN, define the state of the system. Transitions represent sets of related events. Transitions fire (are executed) to change the state of the system. Color sets associated with the transitions indicate the different ways (events) in which the transitions can fire (Arjona & Bueno, 2003).

Given a CPN net structure and an initial marking, the dynamics of a CPN is determined by the enabling and occurrence rules, modeled by arc inscriptions and guards of transitions (Gordon and Billington, 1998) (Zhang et al., 2001). There are two types of arcs in respect to transitions: incoming arcs and outgoing arcs. Incoming arcs connect from input places to a transition, and outgoing arcs connect from a transition to output places. Inscriptions on incoming arcs specify the numbers and colors of tokens from the input places that must be in place in order for a transition to be enabled; and, once the transition occurs, to be consumed. Inscriptions on outgoing arcs specify tokens that the occurrence of a transition produces and puts into the output places (Zhang et al., 2001).

Transition guards are optional, and if present, impose additional conditions for transitions to be enabled and to occur. Enabled transitions can occur either concurrently or sequentially, and in various orders, depending on the numbers of available tokens in places. It is also possible that the occurrence of some transitions changes the state of a net such that the conditions of other enabled transitions are no longer met (Zhang et al., 2001).

Despite its conceptual simplicity, CPNs have proved to be a very powerful modeling tool for discrete event systems, capable of modeling sequences, conflicts, concurrences, and synchronization. They provide a formal framework for the design, specification, validation, and verification of discrete systems (Arjona and Bueno, 2003).

PN has been used in a large variety of areas. Their application ranges from informal to formal systems and from software to hardware systems and from sequential to concurrent systems. As mentioned in (Jensen, 1997b) PN are used in communication protocols, distributed algorithms, computer architecture, computer organization, human-machine interaction and many others areas (Elkoutbi and Keller, 1998).

One issue with using PN to model a manufacturing system is that the difficulty of building and analyzing a PN increases greatly with the complexity of the system being modeled. If a system model is very complex, containing thousands of nodes and transitions, the analysis of this model will be very difficult and time consuming. An approach must be developed where correct PN models of a complex system can be developed and extended from simpler models that are easy to prove valid (Qiao et al., 2002).

The wish to model and analyze large systems by means of PN has shown the need for a modular approach. The main advantages which are expected from this approach are (Sibertin-Blanc, 1993): (a) a better command of the complexity of systems, thanks to a rigorous structure of the model and the possibility to consider different parts of the model independently of each other; and (b) a greater ease to adapt, correct, analyze or reuse a model, thanks to the localization of these tasks at the level of components.

PNs may be connected by merging of transitions, by merging of places, and also by arcs (Sibertin-Blanc, 1993) (Lakos, 2005) (Jiao et al., 2005) (Westergaard, 2004). Transitions merging have the advantage to ease the system's analysis, because many properties are preserved when nets are composed in this way (Sibertin-Blanc, 1993).

Composition of nets by places merging corresponds to communication by variable sharing. The management of the shared place requires an additional synchronization between the nets. Connecting nets by arcs ensures the smallest coupling and it corresponds to communication by "message sending" (arc from a transition to a place) and "message taking" (arc from a place to a transition) (Sibertin-Blanc, 1993).

This work uses the CPN and CPNTools application for modeling the control logic of supervisory system and robots.

4. VALIDATION OF FLEXIBLE ROBOTIC CELLS

4.1. Proposed Approach

The approach proposed in this study for the validation of flexible robotic cells is organized in the following steps:

Step 1: Process Modeling in PFS
The PFS (Production Flow Schema) modeling technique is used to develop a high-level model of the process to be executed in the robotic cell. The model includes the sequence of operations performed by each robot and their coordination by a supervisory system in a high level of abstraction.

Step 2: Definition of Communication Protocols
Using information from the previous step, a high-level protocol for communication between the supervisory system and the robots is defined. It should include the specification of commands and variables that must be exchanged between them in order to perform the process defined in Step 1.

Step 3: Modeling of the Programs in Petri Net
The top-down approach is adopted in this step to generate a Petri net model from the PFS model of Step 1. The programs of robots and supervisory system are detailed to a level such that it includes the exchanging messages defined in Step 2.

Step 4: Validation of Models
The Petri net models of the robots and supervisory system are integrated and the behavior of robotic cell is validated by simulation and/or by a formal verification.

Step 5: Conversion of the Models
The models of robot programs are converted from Petri net to the programming language of the robot simulator. The model of the supervisory system should be converted to an appropriate programming language. This language could be a structured programming language, such as C, a PLC programming language, or other language used in industrial environment, such as LabView.

Step 6: Simulation in 3D
The robot simulator and the supervisory system program must be integrated in order validate the procedure in a 3D environment.

4.2. Case Study

The drilling and placement of rivets in aircraft fuselages are currently implemented in a manner predominantly manual in the Brazilian aircraft industry. Aimed at automating the process, the proposed approach is applied to the design of a robotic cell for assembly of aircraft structural.

The application considered deals with the movement of two robots in tightly coupled tasks, running the drilling and placement of rivets in a longitudinal junction of fuselage, as indicated by the arrows in Figure 1.

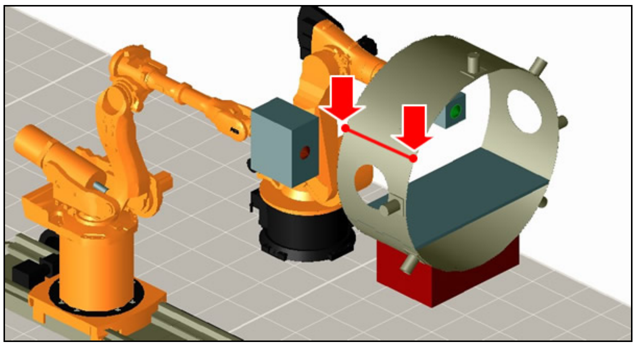

Figure 1. Robotic cell.

To simulate the performance of two industrial robots interacting with the section of fuselage and other components, ROBCAD[1] is used as tool for the graphical simulation.

Following the proposed approach, the supervisory system and the programs of the robots are modeled in a Coloured Petri Net and later converted to a programming language of ROBCAD, the TDL. Information on the TDL can be found at (Siemens PLM Software, 2009). The C programming language is adopted for the supervisory system programming.

The implementation of the proposed procedure in this case study is shown below.

Step 1: Process Modeling in PFS

The process considered in this application is the implementation of a longitudinal junction between two sections of fuselage. The robot 1 uses the tool for drilling and insertion of rivets, while the robot 2 operates as counterpoint of the operation. The PFS of the process is illustrated in Figure 2.

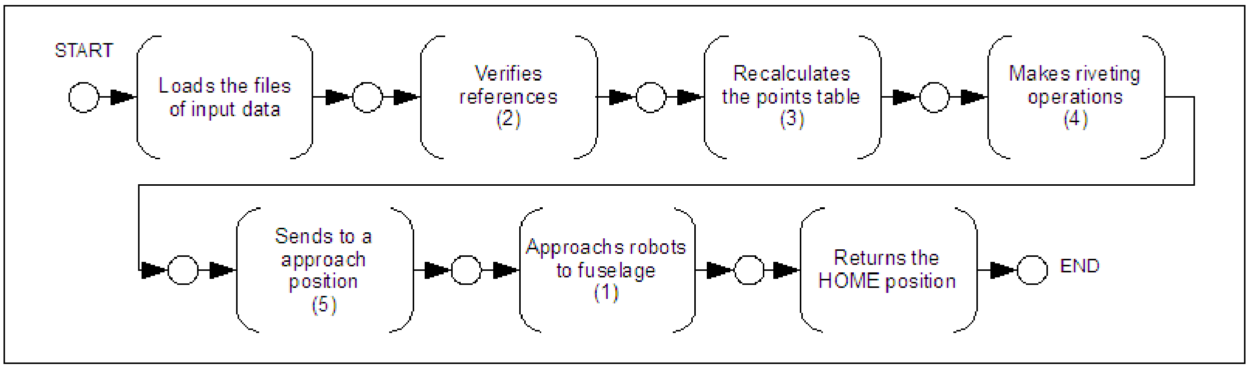
START
Loads the files of input data
Verifies references (2)
Recalculates the points table (3)
Makes riveting operations (4)
Sends to a approach position (5)
Approachs robots to fuselage (1)
Returns the HOME position
END

Figure 2. General PFS of the process.

The process begins with loading of the input data file. This file consists of the positions of the references already installed in the fuselage and the expected positions of the fasteners to be installed.

Following, the robots are closed to fuselage (activity 1). This movement is carried out in fast speed.

The actual position of each reference should be verified by Robot 1 (activity 2), which has a system of vision. In the process used as an example there are only two references.

Depending on the difference between the actual position and the position defined in the data file of the two references, the positions of the rivets are corrected (activity 3). Both robots are then used to make the installation of the rivets (activity 4). Activities 3 and 4 are performed at low speed. Following these activities the robots are sent to the HOME position with command of fast speed.

Step 2: Definition of Communication Protocols

The commands to be exchanged between robots and supervisory system for accomplishing this process are:

- Command "move $joint" to move the robot in fast speed;
- Command "move $linear" to make incremental movements in low speed.

The data to be exchanged between each robot and the supervisory system are organized in the following variables:

- Position and orientation of destination: set of 6 real variables;
- Flag_interno: integer variable that indicates the robot that it should execute a new command in accordance with data of other variables. This variable also identifies the command (joint or linear) to be executed by the robot;
- Flag_externo: binary variable that informs to the system of supervision that the robot completed the last command requested.

Step 3: Modeling of the Programs in Petri Net

The PFS of Figure 3 is detailed in Colored Petri nets. As example, Figure 3 shows the net equivalent to activity 1. The other activities are detailed in a similar way.

[1] The environment ROBCAD software, version 8, supplied by Siemens PLM Software to the Centro de Competência em Manufatura (CCM/ITA).

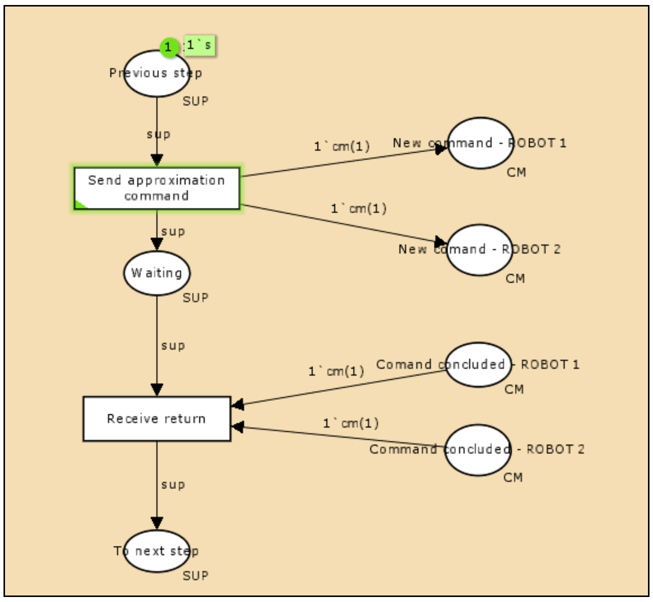

Figure 3. Details of the activity 1.

The nets corresponding to the robot programming are also built. In this case, both networks of Robot 1 and Robot 2 are similar (Figure 4).

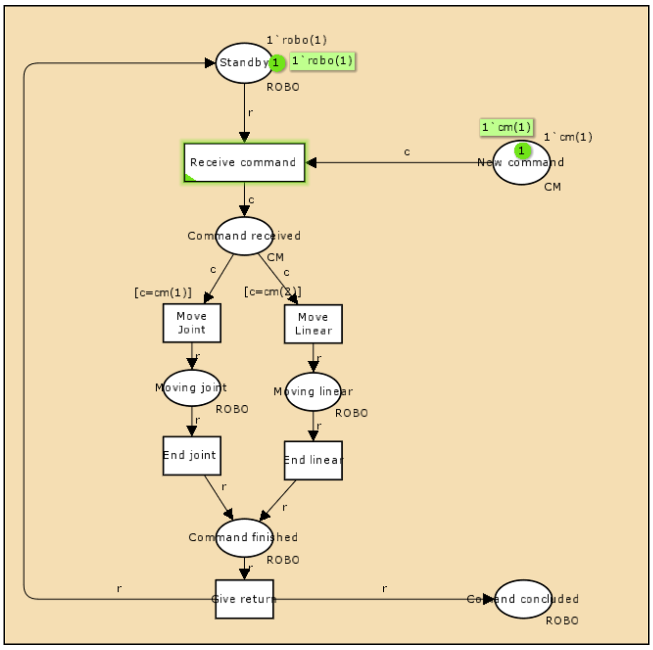

Figure 4. Net of Robot 1.

Step 4: Validation of Model

The models in the Petri nets are validated by simulation. In this case the CPN Tools application is used (CPN Group, 2009).

Step 5: Conversion of the models

The Petri net of the supervisory system is converted into C language. The Petri net corresponding to the robot programming is converted into TDL language of the ROBCAD. The communication between the ROBCAD and the supervisory system program is performed by means of writing and reading of files containing shared variables exchanged between the two applications. Due to the limitation of ROBCAD communication, this is the chosen way to exchange messages with external programs.

The input data are composed by:

- Six real numbers that indicate the position of the first reference with regard to the global coordinate system. The first three numbers indicate the coordinates X, Y and Z and the last three indicate the angles of rotation rX, rY and rZ;
- Six real numbers that indicate the position of the last reference with regard to the global coordinate system. The first three numbers indicate the coordinates X, Y and Z and the last three indicate the angles of rotation rX, rY and rZ;
- One integer number that indicates how many rivets should be inserted between the two references;
- A table with the original position of installation of the rivets.

Step 6: Simulation

In order to verify the process of modeling and creation of the programs, a simulation is implemented using the models developed in the ROBCAD and integrated to the supervisory system written in C language.

Some files are generated during the simulation. They made the communication between the two programs by writing and reading activities, which are listed below:

- "flag1_interno.robcad" and "flag2_interno.robcad": files responsible for receiving the signal of the type of movement from the supervisory system and to provide to robots for reading. Each robot has a file for reading;
- "flag1_externo.robcad" and "flag2_externo.robcad": files responsible for receiving the signal indicating that robots made certain task and to available for reading the supervisory system. These files are only needed when there is the task of synchronization. Each robot has a file for writing;
- "dadosrobo1.robcad" and "dadosrobo2.robcad": files responsible for storing information of coordinate necessary to the movement of robots. Each robot has a file for reading;

To initialize the simulation, a file called "coordenadas.robcad" is created containing the input data for the simulation. Its content is provided in Table 1.

Table 1. Content provided to the supervisory system.

1	5
2	5750 5898.8 2135.05 -105 0 0
3	6750 5898.8 2135.05 -105 0 0

The first line of Table 1 shows how many references should be calculated between the first and last. The second and third rows show the first and last reference relating to the global coordinate system, being six real numbers. On line 1, the X, Y and Z columns indicate respectively the coordinates X, Y and Z and rX, rY and rZ columns indicate the angles of rotation relating to the global coordinate system. In other rows, the X, Y and Z columns indicate respectively the displacements in X, Y and Z and rX, rY and rZ columns indicate the rotations relating to the tool coordinate system.

Table 2. Data calculated by the supervisory system.

Line	Data of the robot 1						Data of the robot 2					
	X	Y	Z	rX	rY	rZ	X	Y	Z	rX	rY	rZ
1	5750.0000	5802.2072	2160.9320	-105.0000	0.0000	0.0000	5750.0000	5995.3924	2109.1681	75.0000	0.0000	0.0000
2	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000
4	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000
6	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000
7	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000
9	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000
10	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000
11	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000
12	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000
13	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000
14	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000
15	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000
16	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000
17	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000
18	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000
19	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000	166.6667	0.0000	0.0000	0.0000	0.0000	0.0000
20	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	100.0000	0.0000	0.0000	0.0000
21	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-100.0000	0.0000	0.0000	0.0000
22	-833.3334	0.0000	0.0000	0.0000	0.0000	0.0000	-833.3334	0.0000	0.0000	0.0000	0.0000	0.0000

The coordinates calculated for the robots by the supervisory system are presented in Table 2. These are the coordinates of the robot 1 from the second to the seventh column and the coordinates of the robot 2 from the eighth to the thirteenth column. The first row is the coordinates of the first reference relating to the global coordinate system. This reference has a distance of approach / departure surface perpendicular to the fuselage, as shown in point 2 of

Figure 5. Thus, the first line provides data to the robots move from the HOME position to the approaching position on high speed and on joint movement. High speed is used because of the low accuracy required. The second line of Table 2 shows the coordinates related to tool coordinate system for the movement of robots at low speed and using linear movement with the goal of approximating the tools to the fuselage in a secure manner. These movements are illustrated in Figure 5, the origin is point 2 and destination is point 3 of the movement of the robot 1. Lines 3 and 4 of Table 2 also show the coordinates relating to the tools coordinate system for departure and displacement to a next approximation. The steps of approximation, departure and displacement are repeated in line 5 to line 22, except the last line because it provides data to movement the robot from last departure point to the first point of approximation, as exemplified the movement of the robot 1 from point 5 to point 2 in Figure 5, thereby ensuring the safety of the cell, because the robots will perform movements known and insurance.

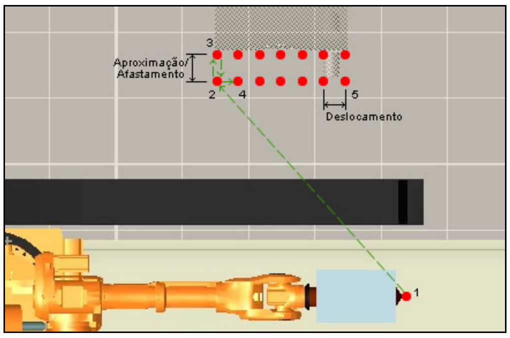

Figure 5. Detail of movement of the robot 1.

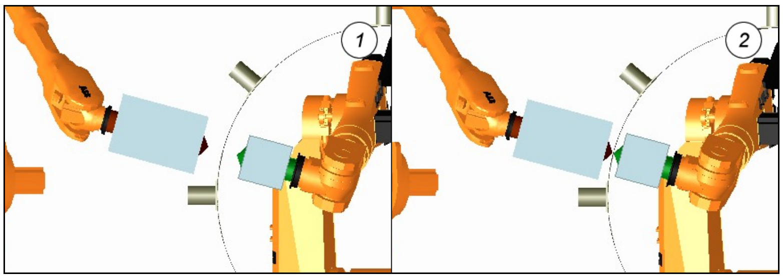

Figure 6. Perpendicularity of the tools relating to the surface of the fuselage.

The last movement from the departure point of the first reference to the HOME position does not require data to the robots.

Figure 6 illustrates the previous (1) and posterior (2) positions for the approximation movement. There are two moments in which the tools are perpendicular to the surface of the fuselage and aligned, moving up slowly and in a linear manner until they are in the fuselage. This provides greater accuracy in the motion.

Results obtained by the graphical simulation with the supervisory program proved to be satisfactory because the access to the fuselage were made without collision and perpendicular, guaranteeing successful of the work. It was observed that the robot 2 came close to the singularity in the last stages of simulation due to a lower range.

5. CONCLUSION AND FUTURE WORKS

Some facilities and benefits that the graphical simulation provides for the design of a robotic cell are observed in this work. Several techniques of access and communication between robots and external systems can be simulated satisfactorily without the need of using real components, reducing costs and increasing safety. Furthermore, research in the area of modeling and simulation is necessary, especially in developing new techniques for modeling.

The modeling procedure proposed in this work using Petri nets provides a better comprehension and detailing of the tasks of each component of the system, characterizing the elements and their relationships. Moreover, the centralization of decision-making and robot coordination by an external element makes the system more flexible and intelligent. This structure leaves the definition of the robot trajectory to the supervisory system, creating a range of possibilities for real time definition of the robot activities, unlike the traditional off-line programming, which fixes and restricts the activities.

The future work is concentrated in the following topics:

- Include the correction of position at each riveting operation when a reference is available;
- Include other steps in the process, such as the conversion of the TDL program to a programming language native to robots, substitution of the message exchanging via simulator by the standard communication protocol used by the robots and the execution of tests in real environment;
- Develop procedures for drilling and placement of rivets in an orbital junction, as illustrated in Figure 7. In this case, the use of track motion as the seventh axis becomes necessary, because it increases the robot accessibility (Aguiar *et al.*, 2007a) (Aguiar *et al.*, 2007b).

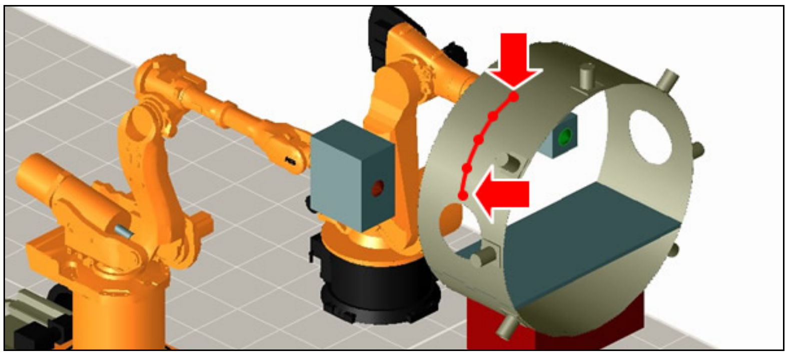

Figure 7. Drilling and placement of rivets of manner orbital.

6. REFERENCES

Aguiar, A.J.C., Silva, A.S.A., Villani, E., 2007a. “Graphic Robot Simulation for the Design of Work Cells in the Aeronautic Industry”, 19th COBEM – International Congress of Mechanical Engineering, Brasília/DF, Brazil.

Aguiar, A.J.C., Silva, A.S.A., Villani, E., 2007b. “Simulação Gráfica de Robôs Aplicada à Indústria Aeronáutica”, VIII SBAI – Simpósio Brasileiro de Automação Inteligente, Florianópolis/SC, Brazil.

Arjona, E., Bueno, G., 2003, “Colored Petri Net Modules of Complex Selecting Criteria”, Simulation, Vol. 79, Issue 7, pp. 410-419, July.

Banerjee, P. and Zetu, D., 2001. “Virtual manufacturing”, New York: John Wiley & Sons. 320p.

Chaimowicz, L., 2002. “Coordenação Dinâmica de Robôs Coopernativos: Uma Abordagem Utilizando Sistemas Híbridos”, Tese de Doutorado, Universidade Federal de Minas Gerais. 126 p., Belo Horizonte, Brazil.

CPN Group, 2009, “Computer Tool for Coloured Petri Nets”. 13 May 2009, <http://wiki.daimi.au.dk/cpntools/cpntools.wiki>.

Elkoutbi, M., Keller, R.K., 1998, “Modeling Interactive Systems with Hierarchical Colored Petri Nets”, Proceedings of the 1998 Advanced Simulation Technologies Conference, p.432-437, Apr.

Gordon, S., Billington, J., 1998, “Applying Coloured Petri Nets and Design/CPN to an Air-to-Air Missile Simulator”, Workshop on Practical Use of Coloured Petri Nets and Design, Aarhus University, Denmark, pp. 1-14, 1998.

Jensen, K., 1997a, “A Brief Introduction to Coloured Petri Nets”, Proceedings of the Third International Workshop on Tools and Algorithms for Construction and Analysis of Systems, pp. 203-208.

Jensen, K., 1997b, “Coloured Petri Nets, Basic concepts, Analysis methods and Pratical Use”. Springer.

Jiao, L., Cheung, T.Y, Lu, W., 2005, “Handling Synchronization Problem in Petri Net-Based System Design by Property-Preserving Transition-Reduction”, The Computer Journal, Vol. 48 No. 6.

Lakos, C.A., 2005, “A Petri Net View of Mobility”, In: IFIP International Federation for Information Processing, pp. 174–188.

Leng, D.Y. and Chen, M., 1997. “Robot trajectory planning using simulation, Robotics & Computer-Integrated Manufacturing”, 13(2): 121-129.

Moore, K.E., Brennan, J.E., 1996, ‘Alpha/SIM Simulation Software Tutorial”, Proceedings of the 1996 Winter Simulation Conference.

Mulyar, N., Van der Aalst, W.M.P., 2005, “Towards a Pattern Language for Colored Petri Nets”, Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2005), volume 576 of DAIMI, pages 39-48, Aarhus, Denmark, October.

Qiao, G., Mclean, C., Riddick, F., 2002, “Simulation System Modeling for Mass Customization Manufacturing”, In: Proceedings of the 2002 Winter Simulation Conference, pp.2031-2036.

Sibertin-Blanc, C., 1993, “A Client-Server Protocol for the Composition of Petri Nets”, Proceedings 14th International Conference on Application and Theory of Petri Nets. Chicago, Illinois, USA, p.377-396.

Siemens PLM Software, 2009. “Robcad Rerefence Manual: Robcad TDL Rerefence Manual”.

Silva, M.F.S., 1996. “Simulação e Programação Off-line de Robôs de Montagem. Porto”, 219 f.. Dissertação (Mestrado em Engenharia Eletrotécnica e de Computadores) - Faculdade de Engenharia, Universidade do Porto.

Silva, M.S., 2004, Aplicação da Simulação à Programação Off-Line de Robôs Industriais, Ingenium, No. 81, pp. 72-77.

Souza, M.C.F., Porto, A. J. V., R., C. A. and Batocchio, A., 2002. “Manufatura Virtual: Conceituação e Desafios”. GESTÃO & PRODUÇÃO, v.9, n.3, p.297-312.

Stobart, R.K. and Dailly, C., 1985. “The use of simulation in the off-line programming of robots”, IEE Control Engineering - Robots and automated manufacture, Series 28, Stevenage, United Kingdom, pp. 11-28.

Wave Report, 2002. “3D – Points to Ponder”. 2 jul. 2007, <http://www.wave-report.com>.

Westergaard, M., 2004, “Towards A High-Level Petri Net Type Definition”, Proceedings of Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets. Satellite event at the 25th International Conference on Application and Theory of Petri Net, Bologna. Italy, June.

Zhang, L., Mitchell, B., Falzon, L., Davies, M., 2001, “Model-based Operational Planning Using Coloured Petri Nets”, 6th International Command and Control Research and Technology Symposium, pp. 1-15. 19-21, Annapolis, MD, USA, June.

Zimmermann, S.C., 2007. “Fábrica Digital”, II Seminário de Manufatura Automotiva, São José dos Campos, Brazil.

7. RESPONSIBILITY NOTICE