ROBOTIC CELL INTEGRATION FOR SCREWING OF AIRCRAFT WING PANELS

GUSTAVO M. CARVALHO, KLEBER R. S. SANTOS, EMILIA VILLANI

CENTRO DE COMPETÊNCIA EM MANUFATURA (CCM), INSTITUTO TECNOLÓGICO DE AERONÁUTICA (ITA)
PÇ. MARECHAL EDUARDO GOMES, 50, 12228-900, SÃO JOSÉ DOS CAMPOS (SP)
E-MAILS: GUSTAVO_CARVALHO@CCM-ITA.ORG.BR, KLEBERSSANTOS@GMAIL.COM, EVILLANI@ITA.BR

Abstract— The screwing of fasteners in the aeronautical industry has strict specifications and quality control requirements. The feasibility of automating these processes represents a major challenge for aircraft manufacturing and contributes to a competitive market. In this work, we present a solution based on the KUKA collaborative robot, LBR IIWA 14 R820, which is current under development at the Aeronautics Institute of Technology (ITA) within the Center for Manufacturing Competence (CCM). The focus of this paper is on the integration and coordination of the equipment that compose the manufacturing cell. A computer vision system is attached to the end-effector to identify the position of the holes to be screwed. A perpendicularity system, composed of a laser sensor, is used to ensure that the perpendicularity error of the robot is within the accepted margin during the application. The algorithm for localizing holes, correcting the position and orientation of the robot, and performing the screwing operation emerges from the integration of all the components. It is first verified in a simulation environment based on the system modelling as a network of timed automata. We then implement and test it in a demonstrator that emulates the process to be performed in an aircraft wing. This work is related to the AME-ASA Project, under development at ITA, in partnership with Embraer and supported by FINEP.

Keywords— Automata, screwing, Iiwa robot, vision system, automation.

Resumo— O parafusamento e fixação de prendedores na indústria aeronáutica possui especificações e requisitos de controle de qualidade rigorosos. A viabilidade de automatizar esses processos representa um grande desafio para a fabricação de aeronaves e contribui para um mercado competitivo. Neste trabalho, apresenta-se uma solução baseada no robô colaborativo KUKA, LBR IIWA 14 R820, que está em desenvolvimento no Instituto Tecnológico de Aeronáutica (ITA) dentro do Centro de Competência de Manufatura (CCM). O foco deste trabalho é a integração e coordenação dos equipamentos que compõem a célula de manufatura. Um sistema de visão computacional é acoplado ao efetuador para identificar a posição dos furos a serem aparafusados. Um sistema de perpendicularidade, composto de um sensor a laser, é usado para garantir que o erro de perpendicularidade do robô esteja dentro da margem aceita durante a aplicação. O algoritmo para localizar furos, corrigir a posição e orientação do robô e realizar o parafusamento é primeiro verificado em um ambiente de simulação baseado na modelagem do sistema como uma rede de autômatos temporizados. Em seguida, implementou-se e testou-se em um demonstrador que emula o processo a ser executado em uma asa de aeronave. Este trabalho está relacionado ao Projeto AME-ASA, em desenvolvimento no ITA, em parceria com a Embraer e apoiado pela FINEP.

Palavras-chave— Autômatos, parafusamento, robô Iiwa, sistema de visão, automação.

1 Introduction

The demand for automation in aviation industry has increased considerably in recent years, encompassing from in-flight applications to manufacturing processes (Kihlman, 2005).

Often, automation arises due to issues related to quality of manual processes and working conditions of human operators. In many cases, automation is also capable of ensuring a more efficient process and promoting a much more productive solution than the ones performed by the human operators.

In order to tackle the challenges of aircraft manufacturing automation, the Centre of Competence in Manufacturing (CCM) proposed first the AME Project, for the automation of the fuselage assembly processes, and now the AME-ASA Project, for the automation of the structural assembly of wings. The AME acronym comes from the name of the project in Portuguese: Automação da Montagem Estrutural, or automation of structural assembly, while ASA translates to wing. Both projects were proposed in partnership with EMBRAER and with financial support from FINEP. Both projects focus on the so-called one-up-assembly, where the product must be assembled in a single step, without removal of components for deburring, cleaning, sealing, in order to maximize efficiency and justify the high cost of automation (DeVlieg & Feikert, 2008).

Considering the context of these projects, this paper presents an automated solution for inserting screws in aircraft wings, considering a flexible programming strategy, in line with Industry 4.0 concepts. The solution considers a scenario where the position of the holes is not previously programmed as part of the robot path. The robot has to scan a certain surface, automatically identify the position and angle of the holes and proposes a sequence of movements for performing the screwing operation. It must also correct its position, as it usually does not provide the necessary accuracy requested by aircraft industry (Cibil, 2006).

Particularly, this paper focuses on the challenges of integrating and coordinating the cell equipment. For this purpose, we modelled the system using automata and verified the behaviour that arose from the equipment integration in the UPPAAL model checking tool.

The main advantage of the proposed solution is the automatic adaptation of the robot to different geometries, without the need of reprogramming. This is a desirable feature, given the low production volume, wide variety and low repeatability that are typical of aircraft industry (Furtado, 2011).

Automation of aircraft assembly processes, such as the one proposed in this paper, reduces recurring costs due to the lack of standardization, and leads to better process quality and traceability. Manual processes are usually strongly dependent on the operators' skills. Furthermore, automation is particularly attractive in the case of regions of difficult access, which exposes the human operator to position considered not ergometric.

The organization of this paper is as follows. Chapter 2 presents the main components of the proposed automation solution, including robot and measurement sensors. Chapter 3 presents and details the software solution, including logical communication among cell components and functionalities provided by each one. Chapter 4 presents the automata model developed to verify the process integration. Finally, in Chapter 5, we discuss the conclusions obtained so far and address future work.

2 The Hardware Solution

The hardware solution is illustrated in Figure 1. Its main components are robot, camera, and laser sensor.

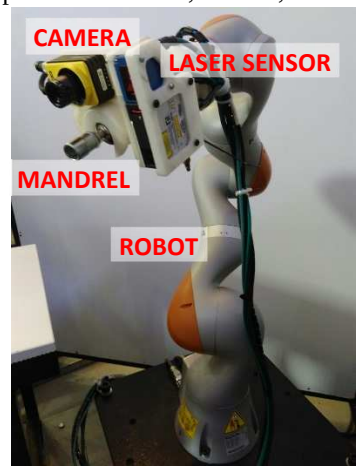

Figure 1- Hardware solution.

2.1 Robot

The robot chosen to be used in the automated solution is the KUKA LBR IIWA 14, a collaborative robot able to perform several movements keeping the end-effector in the same position, due to a redundant joint present in its kinematic chain. The presence of this redundant joint, a seventh degree of freedom, allows the robot to perform complex movements and reach areas of difficult access. This robot is also equipped with sensors in each joint that enable it to feel the force applied on the joints (KUKA Roboter, 2015).

Another interesting feature of IIWA is the availability of a programming mode, known as Impedance Mode, in which the robot behaves as a mass-spring-damper system, allowing its end-effector to perform movements that are typical of a human being, such as when inserting a screw.

2.1 End-effector

One of the requirements of the proposed solution is that it must be capable of automatically determining the position and orientation of holes for screwing.

For this purpose, the robotic end-effector must be equipped with measurement systems. In our solution, it integrates the following modules:

- A vision module that uses a Cognex camera (Cognex, 2018), which generates the X and Y coordinates of each target;
- A laser sensor from Wenglor (Wenglor, 2018) used to guarantee the perpendicularity of the robot, as well as ensure the reading of the Z coordinate when localizing a point of operation;
- A module with mandrel for screwing, with no embedded intelligence.

The Cognex camera has its own programming environment. It provides data to a supervisory system using a set of communication protocols that can be configured in the programming environment (Figure 2). The camera interface is discussed further in Chapter 3.

The Wenglor sensor is provided with Ethernet/IP communication and can be integrated directly with the cell equipment. Additionally, a web server is used for configuration and visualization of the readings in real time, using a web page (Figure 3).

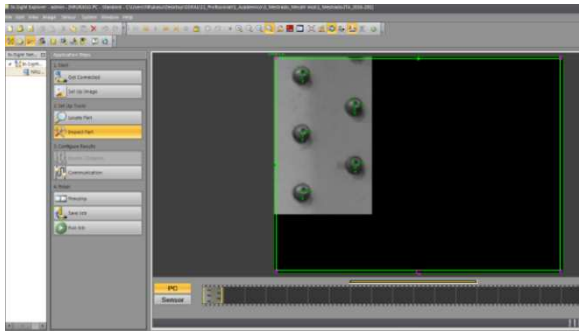

Figure 2 - Cognex in-sight software for identifying fasteners and holes on the wing surface.

Figure 3 - Wenglor sensor web server.

The end-effector was first designed in a CAD environment. A prototype was then built using a 3D printer. Its current version integrates the camera and the sensors, and the screw module.

3 The Software Solution

The behaviour of the proposed manufacturing cell arises from the integration of all components. They are: 1) the supervisory system, implemented in a common PC, 2) the Iiwa robot, 3) the camera, 4) the laser sensor, and 5) the screw module.

The logical integration of the cell is performed via a TCP/IP switch, which connects and allows the exchange of data among any of the following components: supervisory system, robot, camera, and laser sensor. The communication with the screw module is performed via camera, once that it is connected to the I/O module of the Cognex camera. This architecture is illustrated in Figure 4.

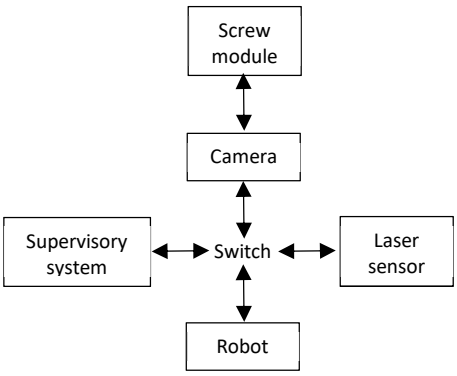

Figure 4 – Communication architecture.

This section describes the functionalities provided by each component and how they interact with the other cell components.

3.1 The supervisory system

The supervisory system coordinates all cell equipment. It integrates information from the Iiwa controller and the camera controller. It also interacts directly with the Wenglor sensor.

The supervisory system is also in charge of establishing and monitoring communication with all equipment. In case of failure, it is responsible to abort the process.

The supervisory system was developed in LabVIEW, an engineering software designed for applications that require testing, measurement, and control, with easy access to hardware from different manufacturers (Bell et al, 2004). It provides a graphical programming interface, including the design of friendly Graphical User Interface (GUI) (Whitley, & Blackwell, 2001).

The process executed by the supervisory system is illustrated in Figure 5.

First, the program initializes all global system variables and then open the communication with each component. Once the communications are validated, an internal state machine is initialized. In its initial state, it performs the perpendicularity routine, which aims at maintaining the end-effector of the robot perpendicular to the work surface, where the holes to be screwed are located. For this purpose, the supervisory system must interact with both the robot and the Wenglor sensor.

After validating the perpendicularity between the end-effector and the surface, the state machine goes to the next state: pattern recognition by the computer vision system. For this purpose, it first position the robot and then communicates with the camera, which is responsible for image acquisition and processing. The supervisory system waits until the camera returns the X and Y coordinates of the all holes found in the work surface.

Once the holes positions are determined, the supervisory system runs an internal routine that aims at organizing the data obtained by the camera software and defining the processing sequence of the holes. It then requests to the robot to move to next hole and performs the screwing operation. This sequence is repeated until there are no more points to process. Then, the supervisory system is responsible for closing the communication with all components.

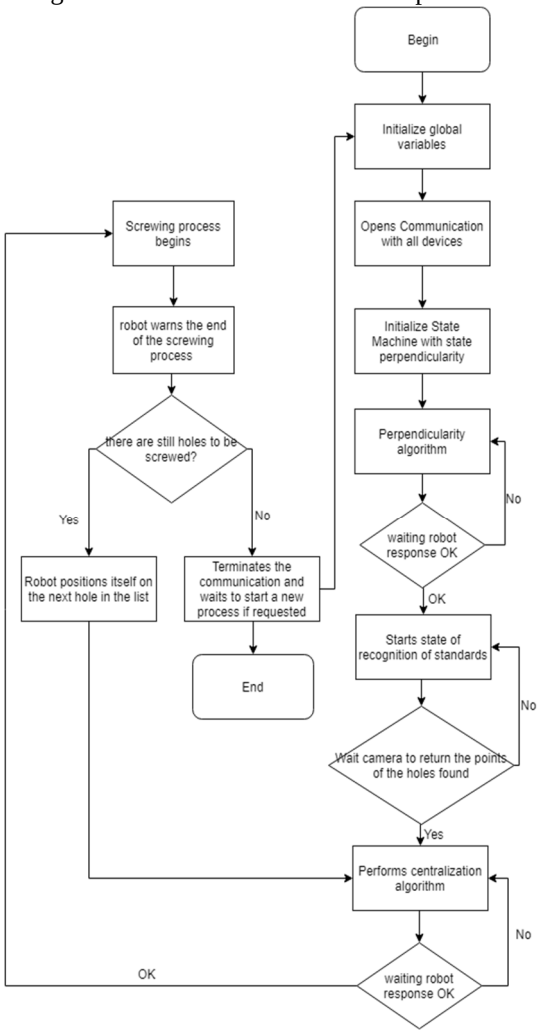

Figure 5 - Process of the supervisory system.

3.2 The camera software

Cognex camera has a proprietary in-sight programming interface. It allows the entire computer vision algorithm to be processed within the camera,

reducing computational time, as it provides dedicated and optimized routines.

The programming environment, called EasyBuilder, has an intuitive interface, allowing users to configure vision applications quickly and easily (Figure 6). All the program parameters and routines are stored in a spreadsheet, which could also be directly edited, as illustrated in Figure 7 (Cognex, 2018).

In this work, the camera provides routines with the following objectives:

- A routine to capture an image and determine the position of all the holes in the image;
- A routine to capture an image and determine the position error, i.e., the distance between the centre of the hole and the centre of the image;
- A routine to start the screwing operation, once that, for convenience, the screw module is connected to the I/O module of the camera (CIO-MICRO).

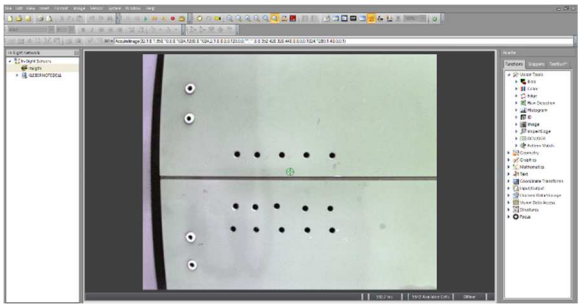

Figure 6 - In-sight programming interface of EasyBuilder.

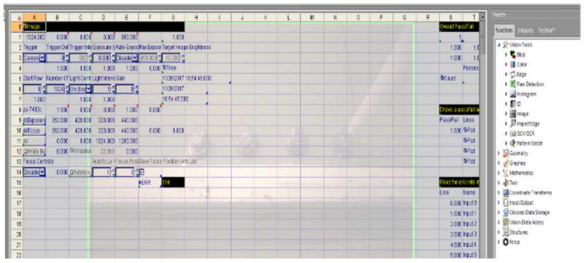

Figure 7 - In-sight programming spreadsheet.

3.3 LBR IIWA 14 Collaborative Robot

The KUKA Iiwa robot programming environment is the Sunrise Workbench (KUKA Roboter, 2015). This software is based on the programming environment Eclipse (Eclipse, 2018). Just like in Eclipse, the Sunrise Workbench is based on the Java programming language, allowing the robot a much higher implementation capacity when compared to other lower-level programming languages used in other KUKA robots. Among the advantages of the Java environment that is important to our application is the extensive library of communication routines, including support to TCP/IP protocols such as HTTP and FTP.

In this project, the robot provides the following functionalities:

- It receives and executes a command to move to a given position;
- It coordinates the perpendicularity routine, which consists of moving the robot to different positions and orientations and acquiring the measurement from the laser sensor (which is done via the supervisory system, for convenience), in order to determine the perpendicularity error. If necessary, the process can be repeated until the error is lower than the tolerance;
- It coordinates the process of correcting the robot position in front of a screwing point and, when the error is under the allowed tolerance, requesting the screwing operation (which is done via camera).

3.2 The laser sensor and the screwing module

Both the laser sensor and the screwing module does not have a significant degree of autonomy or embedded intelligence. They are responsive components that performs operations based on the requests from other components. The laser provides the measured data, while the screw module performs the screwing operation when requested.

4 Verification of system integration in UPPAAL

In order to verify the proposed solution to integrate the cell equipment, this chapter presents its modelling and simulation using synchronous composition of automata and the verification tool UPPAAL.

4.1 Time Automata

Informally, a finite automaton is formed by a set of states and by rules that determine the transitions by states based on the input symbol (Ginfri, 2013). Formally a finite deterministic automaton is a quintuple $(Q, \Sigma, \delta, q_0, q_m)$, where:

- Q: is finite set of states;
- Σ: is an alphabet with a finite set of events;
- $\delta: Q\ x\ \Sigma \rightarrow Q$ is a (potentially partial) transition function that indicates the next state after the occurrence of an event;
- q_0: is the initial state, $q_0 \in Q$;
- q_m: is a set of final (or marked) states, $q_m \sqsubseteq Q$.

In order to model time intervals, timed automata extend this definition and includes the following items (Alur & Dill, 1994):

- Clocks, which are variables that model time evolution;
- Guards, conditions upon the value of clocks that are associate to transitions and must be true for the transition to occur;

- Updates, actions performed by transitions when they fire that can modify the value of clocks;
- Place invariants, conditions upon the value of clocks that are associated to states and must be true in order to the automata remain in the state.

In this work, we decided to model the automated solution as a network of automata, which can be submitted to synchronous composition. When modelling the system in a single automaton, problems may arise due to the explosion of the number of states and transitions, especially when each component has some embedded intelligence and can evolves autonomous from the other components. Any changes in the model would require the creation of a new model from scratch.

In order to tackle this issue, synchronous composition was proposed, so that when adding, modifying or removing components only the corresponding model is affected. In synchronous composition, different automata communicate through shared transitions. When a shared transition fires, it happens in both automata.

The UPPAAL tool uses an approach similar to synchronous composition. It provides binary synchronisation channels. In a channel, a transition labelled with c! synchronises with another labelled c?. When two automata synchronize on channel c, this means that a c! transition of one automaton occurs simultaneously with a c? transition of another automaton. A c! or c? transition can never occur on its own. Only one pair of transition can synchronize at a time. If more than one transition is labelled with c? and are enabled, it is a non-deterministic choice.

4.3 Modelling of the automaton of the screwing process

The software chosen to model our robotic cell is UPPAAL. The tool is designed to verify systems that can be modelled as networks of timed automata extended with integer variables, structured data types, and channel synchronisation. UPPAAL was developed in collaboration between the Real-Time Systems Analysis and Design group at the University of Uppsala, Sweden, and the Basic Computer Science Research at the University of Aalborg, Denmark (Behrmann, 2005).

An automaton model was built for each component of the cell that receives or exchange data with other components. They are:

- Supervisory system (SS);
- Robot (RB);
- Camera (CM);
- Laser sensor (LS);
- Screwing module (SC).

An additional automaton is added to model the user interaction with the system.

Following we detail the module of each component.

The automaton referring to supervisory system is called SS and is presented in Figure 8. It can be organized in four main blocks.

From the stand-by state (SSoff), when the user requests the start of a new process (channel start), the first block, highlighted with an orange dashed line, is responsible for starting and ending communication with other components. For this purpose, it uses a set of channels (or shared transitions): open_RB/close_RB, open_CM/close_CM, open_LS/close_LS. It also uses one clock variable (timer) and associate guards (timer>=time_out) and invariants (timer<=time_out) in order to assure that when at least one component is not available, the process does not continue.

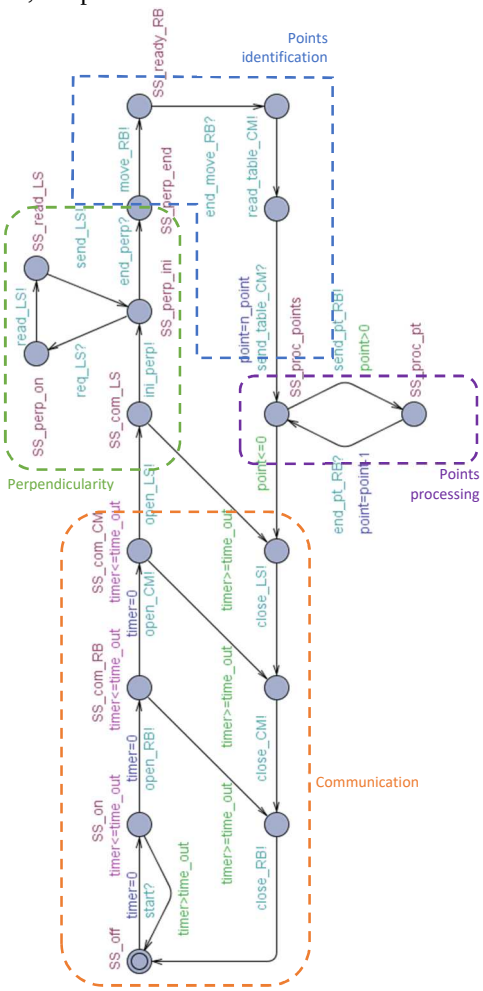

Figure 8 – Supervisory system automaton.

Then, the supervisory system request the robot to execute the perpendicularity routine (channels init_perp and end_perp). Although the routine is coordinated by the robot, for convenience of implementation, the laser sensor is read by the supervisory system (channel read_LS), on the robot request (channels req_LS and send_LS). This block is highlighted with a green dashed line.

Following, the next block is responsible for the identification of the target points. The supervisory system move the robot to the appropriate position (channels move_RB and end_move_RB) and request the

identification of points to the camera (channels read_table_CM and send_table_CM). This block is highlighted with a blue dashed line.

Finally, the supervisory system define the order that the points will be processed and request each operation to the robot (channels send_point_RB and end_pt_RB). This block is highlighted with a purple dashed line.

The robot automaton (RB) is presented in Figure 9. It is composed of four main blocks. The first one (orange dashed line) is the communication with the supervisory system, as presented before. The second one is the perpendicularity routine (green dashed line), which consists of moving the robot to a set of predefined positions and measure the distance to the demonstrator using the laser sensor. At the end of the process, the robot corrects its orientation in order to be perpendicular to the demonstrator.

The purple dashed line identifies the points processing routine, which, under the request of the supervisory system, interacts with the camera (channels calc_error_CM and error_CM) in order to correct the position of the robot. When the positioning error (variable error) is within the tolerance (constant tol), it requires the camera to activate the screwing module (screw_CM and end_screw_CM).

Finally, it provides also a moving routine (blue dashed line) to perform movements under the request of the other components.

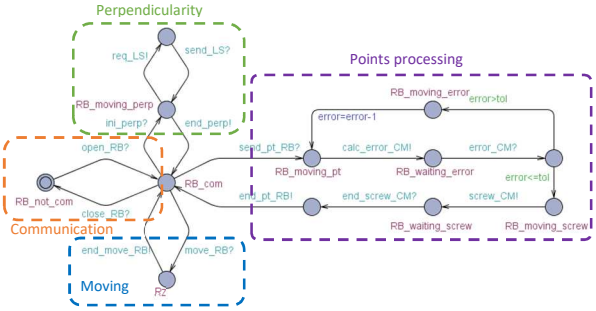

Figure 9 - Robot automaton.

The camera automaton (CM) is presented in Figure 10. It is also composed of four main blocks. The first one is the communication routine (orange dashed line). The blue dashed line marks the camera routine that identifies the set of holes to be processed. The purple dashed line marks the interaction with the screw module (channels ini_SC and end_SC). Finally, the green dashed line marks the camera routine to acquire an image and calculate the centralizing error.

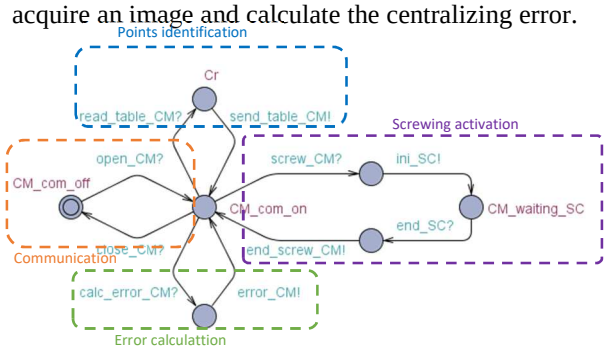

Figure 10 - Camera automaton.

Laser sensor automaton (LS) is presented in Figure 11. It is composed of two blocks: it answers to communication request (orange dashed line) and the provides the sensor data when requested (purple dashed line).

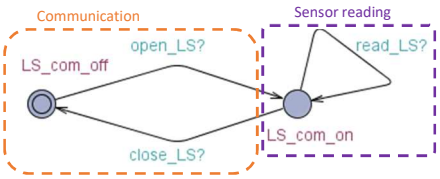

Figure 11 – Laser sensor automaton.

The screw module automaton (SC) is presented in Figure 12. Basically, it responds to the camera commands.

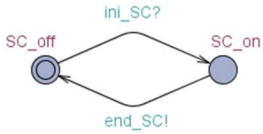

Figure 12 – Screw module automaton.

Finally, an automaton is added to the system to model the interaction with the human operator (Figure 13). It is a simple automaton the starts the process (channel start).

Figure 13 – User automaton.

4.4 Verification

The verification of the automata model is performed using simulation and model checking.

Simulation explores the following ways:

- Random simulation, which explores aleatory sequences of events.
- Simulation of specific scenarios, which corresponds, including the normal sequence of events and scenarios with failures in the communication.

An example of simulation of specific scenarios is presented in Figure 14. It illustrates the communication among the automata until the screwing of the first point, with no communication failure.

Once the model has been debugged using simulation, formal verification is carried out using model checking. In UPPAAL, model checking is performed by specifying the desired properties of the system in CTL (Computational Tree Logic) and then submitting them to the model checker.

Properties can be defined using state formulas and CTL operators. The following operators are available:

- E<> p – there exists a path, where the p will hold sometime in the future.
- E[] p – there exists a path where in every state, p holds.

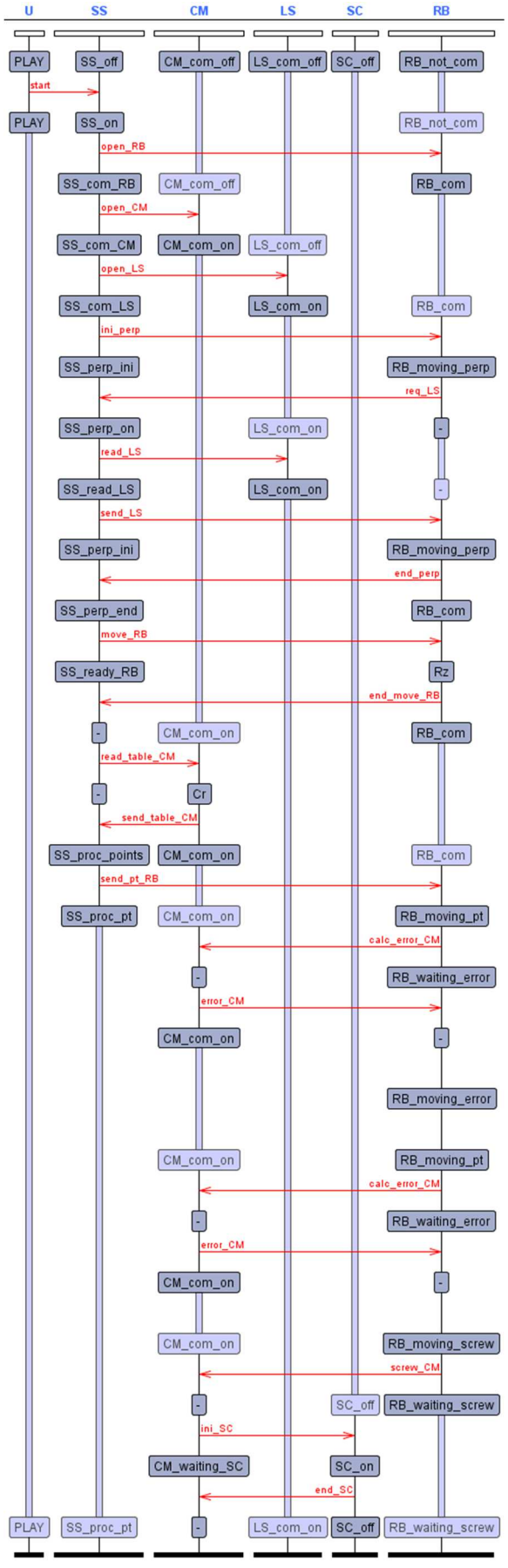

Figure 14 - Simulation of specific scenario.

- A<> p – for all possible paths, p will hold sometime in the future.
- A[] p – for all possible paths, in every state, p holds.

In the case of the integration of the screwing manufacturing cells, example of verified properties:

- Absence of deadlock:

  A[] not deadlock
- Reachability of the automata states, such as the possibility of reaching the state where the points are processed:

  E<> SS.SS_proc_points
- Consistence among the system states. An example is to assure that when the Screw module is performing the screwing operation, all the other modules the robot and the camera are in the appropriate state:

  A[] SC.SC_on imply (CM.CM_waiting_SC and RB.RB_waiting_screw)

The CTL properties are used to specify that whatever happens in the system, the combined evolution of the components preserve consistency.

4.5 Implementation

This section illustrates the implementation of the modules integration in the corresponding programming language of each module. This implementation is performed manually, as each automaton is converted to a different programming language. The consistence between the automaton and its implementation is verified by manual inspection of the code.

We provide here an example of the supervisory system, which is implemented in LabView. In this case, a set of subVIs were developed to communicate and to exchange data with each component. These subVIs are illustrated in Figure 15.

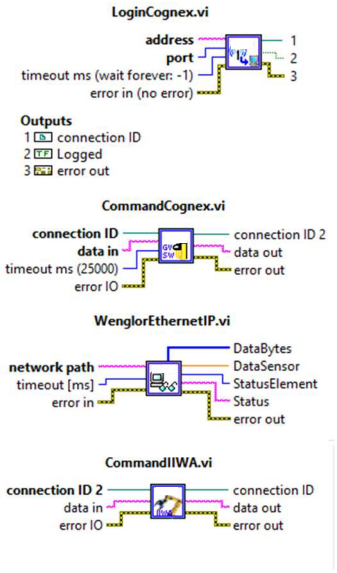

Figure 15 – Communication subVIs in LabView.

Following, Figure 16 illustrates the initialization of the state machine in the LabView and the opening of the communication with the components. In a similar way, communicating routines are introduced in robot and camera programming environment. In the case of the laser sensor, communication is configured through the sensor web server.

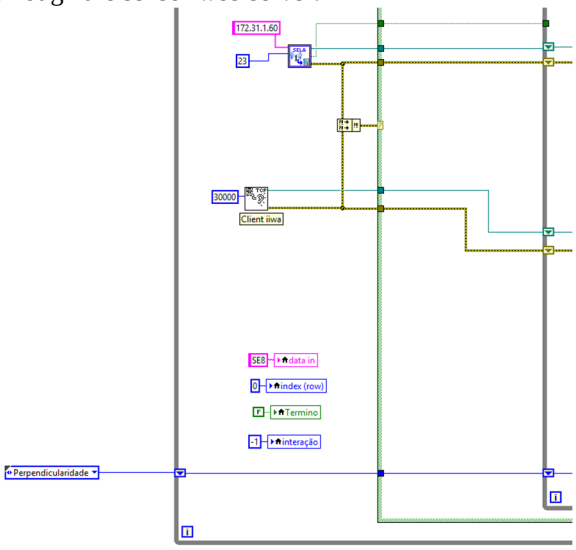

Figure 16 - Startup in LabView.

After implementation, the components have been submitted to tests, both isolated and integrated into the demonstrator. We observe that so far, the tests have been performed without the screwing module, which is under acquisition process.

5 Conclusion

This paper presents the design of a cell to automate the screwing process in aircraft wings. In order to assure the flexibility of the cell to adapt to different products without the need of reprogramming, a measurement system, composed of a camera and a laser sensor, is added to the robotic end-effector.

The integration of the cell equipment is modelled as a network of automata and verified in the UPPAAL tool. The simulation of different scenarios in UPPAAL confirm that the proposed solution is consistent from an integration perspective. All the components of the cell are capable of executing their process cyclically and the coordination of shared data coming from the different components of the cell is working as expected.

Once the cell integration has been verified, we carried out the implementation of the communication among equipment and perform the necessary tests. The use of modelling and simulation as a preliminary step to analyse the cell integration resulted in a limited number of errors and reduced the development time.

The next steps of the project are toward collaborative robotics. In order that the human operator can share the workspace with the Iiwa robot, we must assure the safeness of the cell. Although the robot itself has already been certified for collaborative operation, its integration with other cell equipment must also be verified. For this purpose, we plan to include into the UPPAAL model and software implementations the treatment of the most relevant types of failures that can occur in the cell. Then, we can verify the robustness and safeness of the cell using both simulation and model checking.

Acknowledgements

The authors acknowledge the financial support from Brazilian funding agencies CAPES, CNPq and FINEP.

References

Alur, R; Dill, D.A. (1994). Theory of Timed Automata. Theoretical Computer Science, v.126, pp. 183-235.

Behrmann, G. et al., 2005. A Tutorial on UPPAAL. Proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT‟04). LNCS v. 3185.

Bell, I. et al. (2004). Integration of hardware into the LabVIEW environment for rapid prototyping and the developmento of control design applications. Proceedings of UKACC Control 2004 Mini Symposia, Bath, pp. 79-81.

Cibiel, C. and Prat, P. (2006), “Automation for the Assembly of the Bottom Wing Panels on Stringers for the A320”, *In: SAE World Congress*. Detroit, USA.

Cognex (2018). COGNEX Corporation. [Online] Available at: https://www.cognex.com/ [Acesso em 20 02 2018].

DeVlieg, R. & Feikert, E. (2008). One-Up Assembly with Robots. SAE International, p. 4.

Eclipse (2018). Eclipse. [Online] Available at: https://www.eclipse.org/ [Acesso em 20 2 2018].

Furtado, L. F. F. et al (2011). Comparative study between two methods for perpendicularity corrections in robotic manipulators. In: 21th International Congress of Mechanical Engineering (COBEM), Natal.

Ginfri, L. (2013). Autômatos sincronizados e a Conjectura de Cerny. IME-USP, ed. São Paulo.

Kihlman, H. (2005), Affordable Automation for Airframe Assembly: Development of Key Enabling Technologies”, Thesis (Phd) - Linköpings Universitet, Linköping, Sweden.

KUKA Roboter (2015). Programming and Configuration of LBR iiwa. KUKA Roboter GmbH ed. Augstburg: Pub COLLEGE Workshop LBR iiwa.

Wenglor (2018). Wenglor the Innovative family. [Online] Available at: https://www.wenglor.com/ [Acesso em 20 2 2018].

Whitley, K.N.; Blackwell, A.F. (2001) Visual programming in the wild: a survey of LabVIEW programmers. Journal of Visual Languages and Computing, vol. 12, pp. 435-472.