Università degli Studi di Modena e Reggio Emilia

DOTTORATO DI RICERCA IN
INGEGNERIA DELL'INNOVAZIONE INDUSTRIALE

Ciclo XXVIII

Hardware-in-the-loop (HIL) simulation to improve Virtual Commissioning of Control Software

Candidata: **Rita Tessari**

Direttore della Scuola:
Prof. Mauro Dell'Amico

Relatore:
Prof. Cesare Fantuzzi

Esame finale anno 2016

Reggio Emilia, Italy, March 2016.

Abstract

In this Ph.D. thesis a new HIL environment has been implemented in order to validate virtual prototypes of automatic industrial machines in early stages of the design development, with both cost and time savings.

The novel HIL scheme is applied following a Model-Based Systems Engineering (MBSE) workflow and SysML is the graphical modeling language chosen for the application of the MBSE approach. This mechatronic methodology is implemented via the design of a packaging machine, with relative simulation and validation. Mechanical and control software parts have been developed identifying modular blocks which satisfy the target specifications of the machine. Virtual prototypes are built for each packaging machine module with MATLAB Simulink and SimMechanics. In order to implement the correspondent HIL simulations, Beckhoff TwinCAT software is employed.

Eventually experimental results of Simulink and HIL simulations are presented.

Sommario

In questa tesi di dottorato è stato implementato un nuovo tipo di simulazione HIL, col fine di validare prototipi virtuali di macchine automatiche industriali nelle fasi iniziali dello sviluppo del progetto, con risparmio di costi e di tempo.

Il nuovo schema HIL è stato applicato seguendo un workflow MBSE (Model-Based Systems Engineering) e SysML è il linguaggio di modellazione grafico scelto per applicare l'approccio MBSE. Questo metodo meccatronico è implementato mediante il progetto di una macchina confezionatrice, con relativa simulazione e validazione. Le parti meccaniche ed il software di controllo sono stati sviluppati identificando blocchi modulari che soddisfano le specifiche della macchina. I prototipi virtuali sono stati realizzati per ognuno dei moduli della macchina confezionatrice mediante MATLAB Simulink e SimMechanics. È stato utilizzato il software Beckhoff TwinCAT per implementare le corrispondenti simulazioni HIL.

Infine sono presentati i risultati sperimentali delle simulazioni Simulink e HIL.

Acknowledgements

This thesis is the result of years of work, and I really need to thank who made it possible. I would like to express my deepest thanks to my supervisor, Prof. Cesare Fantuzzi, and to Prof. Cristian Secchi, for giving me this opportunity and for their help, teachings and support during these three years. Many thanks go to Angelo Caputi, Technical support at Beckhoff Automation, for his assistance in troubleshooting with TwinCAT software. I'd like to thank Davide Borghi, Technology Specialist at Tetra Pak Packaging Solutions SpA, for the support to my approach to mechatronic issues. I'd also like to thank all the guys of the ARSControl group. Thanks for everything.

Contents

Chapter 1

Introduction

Mechatronics refer to systems combining mechanical, electronic, control, and software engineering. The challenge of integrating different technologies to create a reliable device is not a trivial task, especially when you consider that the number and complexity of the integrated systems is climbing. In particular, over the last few decades, the proportion of software development involved when designing new machinery and systems has impressively increased.

In the future, businesses will establish global networks that incorporate their machinery, warehousing systems and production facilities in the shape of Cyber-Physical Systems (CPS). In the manufacturing environment, these Cyber-Physical Systems comprise smart machines, storage systems and production facilities capable of autonomously exchanging information, triggering actions and controlling each other independently. This facilitates fundamental improvements to the industrial processes involved in manufacturing, engineering, material usage and supply chain and life cycle management. The Smart Factories that are already beginning to appear employ a completely new approach to production. Smart products are uniquely identifiable, may be located at all times and know their own history, current status and alternative routes to achieving their target state. The embedded manufacturing systems are vertically networked with business processes within factories and enterprises and horizontally connected to dispersed value networks that can be managed in real-time, from the moment an order is placed right through to outbound logistics. In addition, they both enable and require end-to-end engineering across the entire value chain [1].

This is the vision of a profoundly transformed industry landscape, called Industry 4.0, or the fourth industrial revolution, which is already on its way. This time, physical objects are being seamlessly integrated into the information network. The Internet is combining with intelligent machines, systems production and processes to form a sophisticated network. The real world is turning into a huge information system.

Industry plays a central role in the economy of the European Union, accounting for 15% of value added (compared to 12% in the US). It serves as a key driver of research, innovation, productivity, job creation and exports. Industry generates 80% of the EU's innovations and 75% of its exports. Including its effect on services, industry could be considered the social and economic engine of Europe. Yet European industry has lost many manufacturing jobs over the last decade, and is facing tougher competition from emerging markets. The ghost of "deindustrialization" currently haunting European governments and the European Commission is galvanizing them into action [2].

Industry 4.0 will be an answer to the challenges lying ahead. Although some argue that services may eventually replace manufacturing, this is unlikely as the two sectors are closely intertwined. Manufacturing creates value in the service sector.

Innovation, automation and sophisticated processes are at the root of industrial success strategies and have proven to be critical in maintaining a leading position.

Industry 4.0 is based on new and radically changed processes in manufacturing companies: Factory 4.0.

In this concept, data is gathered from suppliers, customers and the company itself and evaluated before being linked up with real production. The latter is increasingly using new technologies such as sensors, 3D printing and next-generation robots. The result: production processes are fine-tuned, adjusted or set up differently in real-time.

Within the modular structured Smart Factories of Industry 4.0, CPS monitor physical processes, create a virtual copy of the physical world and make decentralized decisions. Over the Internet of Things (IoT), CPS communicate and cooperate with each other and humans in real time. Via the Internet of Services (IoS), both internal and crossorganizational services are offered and utilized by participants of the value chain.

The Industry 4.0 design principles, which can be used for the implementing Industry 4.0 scenarios in companies are, the following:

- Interoperability: in Industry 4.0 companies, CPS and humans are connected over the IoT and the IoS. Standards will be a key success factor for communication between CPS of various manufacturers.

- Virtualization: it means that CPS are able to monitor physical processes. These sensor data are linked to virtual plant models and simulation models. Thus, a virtual copy of the physical world is created. In the Smart Factory plant the virtual model includes the condition of all CPS. In case of failure a human can be notified.

- Decentralization: the rising demand for individual products makes it increasingly difficult to control systems centrally. Embedded computers enable CPS to make

decisions on their own. Only in cases of failure tasks are delegated to a higher level.

- Real-Time Capability: for organizational tasks it is necessary that data is collected and analyzed in real time. In the Smart Factory the status of the plant is permanently tracked and analyzed. Thus, the plant can react to the failure of a machine and reroute products to another machine

- Service Orientation: the services of companies, CPS, and humans are available over the IoS and can be utilized by other participants. They can be offered both internally and across company borders. The Smart Factory plant is based on a service oriented architecture. As a result, the product specific process operation can be composed based on the customer specific requirements

- Modularity: modular systems are able to flexibly adapt to changing requirements by replacing or expanding individual modules. Therefore, modular systems can be easily adjusted in case of seasonal fluctuations or changed product characteristics. In the Smart Factory plant, new modules can be added using the Plug&Play principle. Based on standardized software and hardware interfaces, new modules are identified automatically and can be utilized immediately via the IoS.

In this context, engineers tasked with designing mechatronic machines must be prepared with a comprehensive set of requirements and the tenacity to iron out all of the problems that will occur when integrating multi-domain diverse systems.

More and more, simulation is being used to detect and eliminate integration issues. Simulation has been used on individual components and subsystems for quite some time, but the real value becomes apparent when you can integrate those individual systems virtually. The tremendous cost savings of finding problems before you commit to hardware are well-known. What's more important, however, is being able to optimize system performance. That's something you can only achieve in simulation when you can combine those different domains with the controllers in a single simulation environment.

Best-in-class organizations are more likely to use simulation to validate system-level behaviour. Their design process begins with modeling and simulation, and those virtual tests are used to shape the requirements for the system. Increasing or decreasing the sizes of motors, generators, engines, and other components is more cost effective in simulation than using hardware prototypes, and can take the design team down paths that otherwise may have been left unexplored [3].

Many machine builders have developed internal libraries to facilitate modeling, simulation, and multi-domain optimization. In these instances, a core group of engineers develops libraries of models that span electrical, mechanical and thermal systems to be

shared with other teams. Those teams take the components they need, combine them with models of the systems they are designing, and simulate the entire system. By varying the parameter values or using optimization algorithms, those groups can see if the system is meeting design goals. The results of those simulations are fed into the requirements process so that components can be optimally sized.

The breadth of what is considered a mechatronic system is rapidly expanding as new technologies are developed. By running simulations early in the design process, engineers can try out new technologies and see if they will improve the design. Moreover simulations enable optimizing quality and functionality while shortening design iterations and reducing overall development time.

This thesis focuses on improving the engineering process of packaging machine development by using hardware-in-the-loop (HIL) simulation for virtual commissioning of control software.

Nowadays the control units represent a relatively low-cost part of the whole machine system but they are the most critical components. In fact, the control is largely responsible for the performance of the manufacturing production, the safety of system, and the human operators safety. Traditionally, control software can be tested after the construction of the machine, slowing the whole engineering process of development, and wasting resources. These facts have pushed the development of simulation technique to support the controller verification and validation (V&V) in parallel to the machine development. In particular, the HIL (i.e. the connection of the real controller with a simulated model of the machine for V&V) is a promising technique in this area.

The V&V of the controllers for complex mechatronic devices is a key step to assure quality of the overall system.

The objective of this thesis has been to find out a methodology for the development of a HIL simulation, with the aim to the V&V of the distributed control system at early development stage of a packaging machine

A formal Model-Based Systems Engineering design workflow is exploited for a packaging machine design. Modular packaging blocks have been identified in SysML, then modeled in Simulink and in Beckhoff TwinCAT environment they have been simulated in real-time.

Models have been developed implementing closed control loop systems formed of SimMechanics model and control system based on a proportional integral derivative (PID) controllers. After generating the C++ code, the Simulink packaging machine models have then been properly integrated into Beckhoff TwincCAT software in order to perform HIL simulations.

The relative experimental results confirm that these prototypes were effective, since they consistently met system requirements. Preliminary results on the HIL simulation for a Jaw System Unit, a fully tested packaging device, are presented. In order to test the new environment capabilities, also Virtual Machine and External mode versions of the JSU HIL project have been deployed.

In particular, the main contributions of this thesis include:

- A novel HIL scheme deploying just one laptop, even at modular level [4]
- A proof of concept of reusability and reconfigurability by successfully assembling an already available modular block in a new modular packaging machine design project [5]
- Feasibility of integrating many Simulink modular models in a composite model, running on one PLC and one CPU [6]
- Identification of a modular MBSE development workflow for packaging machine design, encompassing HIL simulation, which constitutes a Virtual Commissioning of the control software [4] [5] [6]

The thesis is organized as follows:

Chapter 2 describes the state-of-the-art of hardware-in-the-loop simulation and its key role for the controller verification and validation of complex automatic machines. Then it defines the principles of virtual commissioning and the main features of control for packaging machines. Finally packaging line design issues are described.

Chapter 3, after the description of the main characteristics of the software tools employed in our research activities, presents the development process of a modular mechatronic machine design and the relative software architecture.

Chapter 4 presents the progress of our research work and its application into two case studies. First, it is presented a fully tested device, the JSU, in order to minimize the impact of the new platform. By means of the Left JSU project, the new HIL scheme proposed in this thesis will be validated. With the second case study, this novel HIL scheme will be validated at modular level. The second case study is a packaging machine design developed from scratch, which will provide experimental validation of the whole proposed software architecture and, furthermore, proof of concept of reusability and reconfigurability at modular level.

The present research has been developed inside “Fabbrica Intelligente – ADAPTIVE: A modular and adaptive approach to the design of digital factories”, a project funded by the Italian Ministry for University and Research [7].

Chapter 2

HIL and Virtual Commissioning

2.1 Introduction

There is nothing more difficult than launching a new plant or a new product in an existing plant: hours of adaptations, trials, pre-series testing requiring a high-calibre launch team and numerous unexpected cost overruns. A day lost through a standstill of production means a huge revenue loss for many businesses. Industry 4.0 will use virtual plants and products to prepare the physical production. Every process is first simulated and verified virtually; only once the final solution is ready is the physical mapping done – meaning all software, parameters, numerical matrixes are uploaded into the physical machines controlling the production. Virtual plants can be designed and easily visualized in 3D, as well as how the workers and machines will interact [2].

In this context, virtual simulations are more and more key factors in the mechatronic machine project developments.

Simulation techniques, common practise in fields such as automotive or aerospace, are gaining ground also in the manufacturing industry, where the timing of the release of a new product on the market is increasingly reduced, and where the demand for customized solutions requires the development of many variations of the same machine or production line.

The interaction of heterogeneous control hard and software plays a key role in enabling mechatronic production systems to become flexible and agile systems. Nevertheless, control software engineering still tends to be the last step within the development process. To a large extent it is carried out during the commissioning phase of the production ramp-up. On the first hand this leads to a loss of time and quality, as well as to a loss of reputation and future orders on the second hand. A method that is referred to as Virtual Commissioning tries to overcome this situation. The aim is to enable control software engineering to both take over the initiative in system design and to perform important

activities earlier in the design process of production equipment.

Hardware-in-the-loop (HIL) simulation is a technique that connects the real controller with a simulated model of the machine for the controller verification and validation of complex automatic machines. The complexity of the plant under control is included in the hardware-in-the-loop platform by adding a model representation of all related system dynamics. Therefore hardware-in-the-loop simulation tests both the control system and the plant, supporting the parallel development of the control and the mechanics at early system design stage.

The simulation of a new system can be done in a few weeks instead of months, and without having to physically realize the mechanical parts, if not in the final version.

In this thesis HIL tools are a part of the system development proving strategy of a packaging machine.

2.1.1 Outline

The outline of the Chapter is as follows: Section 2.2 presents the state-of-the-art of hardware-in-the-loop simulation, Section 2.3 defines the principles of Virtual Commissioning, Section 2.4 describes the main features of control for packaging machines, and, finally, Section 2.5 discusses packaging line design issues.

2.2 HIL simulation state-of-the-art

Hardware-in-the-loop simulation is a scheme that incorporates some hardware components of primary concern in the numerical simulation environment [8].

Because HIL simulation is an important verification step in any embedded system, this technique is used widely in a range of fields [9], particularly in the aerospace and automotive industry [10], but even in energy [11], medicine [12], marine [13], rail [14], telecommunications [15], and even other sectors such as manufacturing and automation [16]. Actually control unit algorithms and production sequences have many similarities, whether processing metal or wood, whether printing on textiles, plastic or paper or converting them into packaging, or whether producing or packaging pharmaceuticals, food and beverage or tobacco products.

Over the past three decades the unprecedented rise in the available computing power has led to an inevitable growing use of computer simulations during all phases of mechatronic products development. In parallel with this trend, HIL simulations are increasingly being used as cost effective means for rapid prototyping, integration and validation of complex engineering systems [17].

Since some of the control-loop components are real hardware, HIL simulation is closer to the real process than other simulation methods. What's more, the biggest advantages of HIL simulation are the availability of saving of cost and development time and testing of the effects of faults and failures of actuators, sensors and computers on the overall system, as well as flexibility in optimizing the design parameters. In the end, for primarily designing, verifying and optimizing the actuator and control strategy, HIL simulation test rigs have been proved to be effective platforms [18].

2.2.1 State-of-the-art: from the beginnings

In automotive engineering the history of hardware-in-the-loop simulation goes back to the 1980s. This was first generation of HIL simulation. To begin with, HIL simulation was widely used for testing individual components (e.g., new actuators for active suspensions), mainly in universities and research or advanced engineering departments. However, it was relatively complex and expensive owing to its relatively low-level electronic technology and insufficient functionality to meet the evolving sophistication of the electronics controller. In the 1990s, an increasing number of electronic control units began to be installed in vehicles, and there was greater emphasis on testing these electronic control units, the second generation HIL simulation, which executes analytical and empirical vehicle description and performs extensive I/O operations in real time, was developed. HIL simulation test systems were increasingly used not only in advanced engineering and control design, but also in production development, where they gradually replaced the open-loop or stimulus simulators that were still prevalent at the time. The main focus of these black box tests was on function verification, and on testing self diagnostics and hardware, to support production-level verification by automated regression tests [8].

2.2.2 HIL simulation architecture

HIL simulation enables testing of components under development while communicating with software models that simulate a part or the rest of the system. The System Under Test (SUT) in a HIL setup responds to simulated signals as if they were communicating with the actual real hardware [16].

According to [8] the basic principle of the HIL simulation of an electronic control unit is illustrated in Fig. 2.1. The control system outputs are fed directly to the simulator, where they are sampled and used as input variables. That is the connection of the real controller with a simulated model of the machine [19].

Fig. 2.2 shows a HIL simulation setup typical at the present time: the desktop computer (development hardware) contains the real-time capable model of the controller and

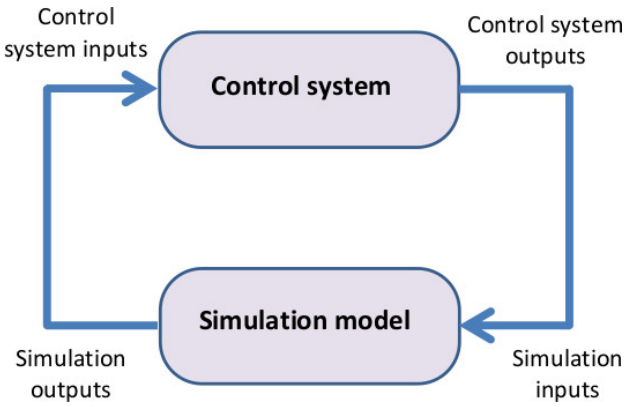

Figure 2.1: Basic principle of HIL simualtion

plant. The development hardware also contains an interface with which to control the virtual input to the plant. The controller hardware contains the controller software that is generated from the controller model. The real-time processor (target hardware) contains code for the physical system that is generated from the plant model [20]. Therefore HIL simulation can test the controller hardware and the performance of generated control code, before hardware prototypes are available.

With reference to a production system, as illustrated in Fig. 2.3, the control unit, or Electronic Control Unit, or ECU, or controller, is a Programmable Logic Controller (PLC). Since a PLC is a dedicated controller, it will only process the program to control machine operation over and over again.

Furthermore, since in the design of a production system the two major activities are mechanical design (device specifications and layouts) and electrical design (device behaviour and system control), the relative HIL scheme provides two major benefits: (1) the verified control program can be directly used for the real plant with minor adaptations; and (2) the verified 3D layout model can be used to generate the detailed drawings for the implementation of the plant [16].

Fig. 2.4 reports the basic elements of an HIL simulator, which can be categorized as simulation environment, input/output (I/O) resources, signal conditioners, hardware components and user interface. The core of simulation environment is a real-time capable simulation processor platform. Furthermore, it includes means of modeling plant components and provides adequate user interfaces. Signal conditioners perform the necessary operations to send/receive signals to/from the controller. Hardware components include all control units to be examined as well as further components like loads, actuators, displays, etc. Finally, the user interface is the medium for communicating information to/from the user or external computers. Activities may include the creation of test scripts, definition of pass/fail criteria, execution of tests, and display of graphical results.

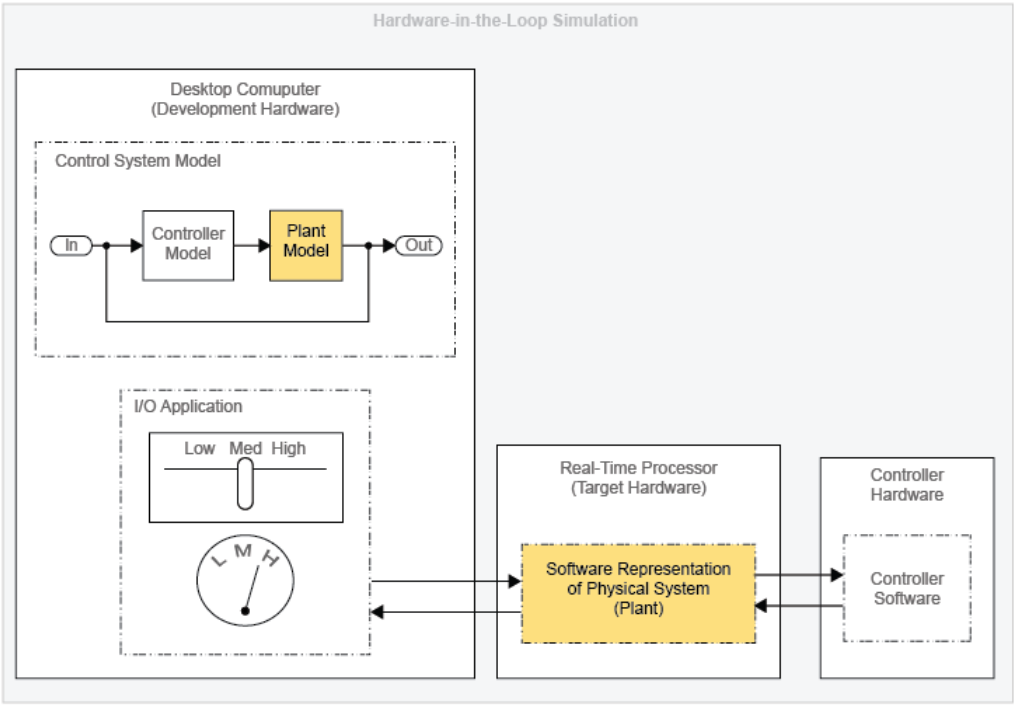

Figure 2.2: Typical at the present time HIL simulation setup

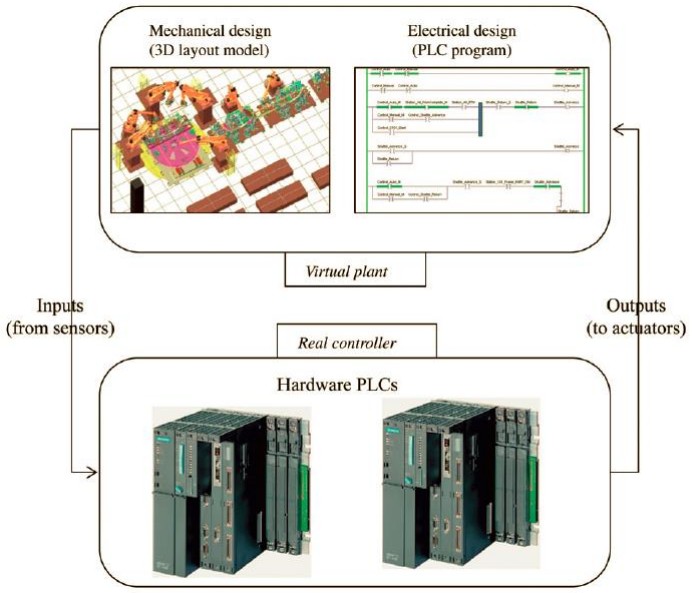

Figure 2.3: HIL architecture for a production system

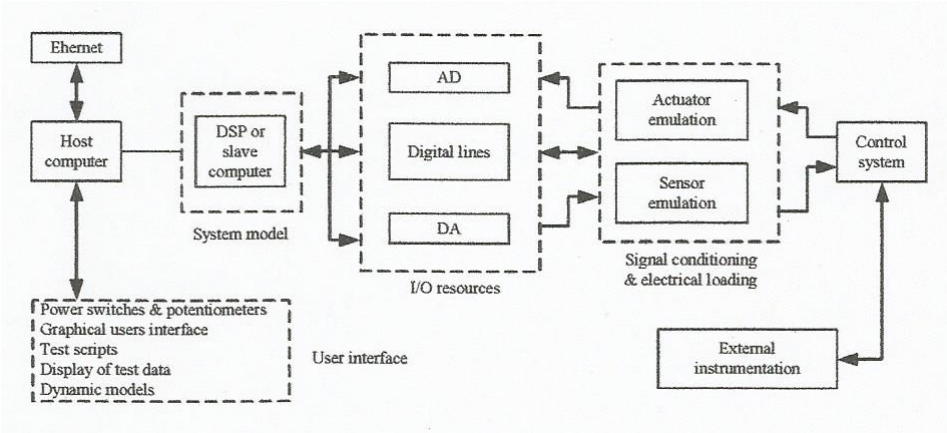

Figure 2.4: Basic elements of a HIL simulator

Together these basic elements provide an environment to repeat test sequences in a precise manner to effectively investigate and document a controller's operation [8]. Actually HIL simulation shows how your controller responds, in real time, to realistic virtual stimuli, testing your controller's response to unusual events. For example, you can model extreme weather conditions like earthquakes or blizzards. You can also test how your controller responds to stimuli that occur in inaccessible environments like deep sea or deep space.

2.2.3 Objectives of HIL

For the development of mechatronic systems, Model-Based System Engineering (MBSE) design through the use of automatic code generation technology and hardware-in-the-loop testing, alleviates errors introduced during manual implementation and realization tasks and shortens the path to product delivery by generating code for testing, calibration, and final production [8].

Fig. 2.5 shows where HIL simulation fits into the MBSE design-to-realization workflow. This is the "V" diagram, which represents the sequential design and validation phases in the traditional development of mechatronic systems [21].

While validation involves using actual plant hardware to test your controller in real-life situations or in environmental proxies (for example, a pressure chamber), in HIL simulation you do not have to use real hardware for your physical system (plant). Further you also do not have to rely on a naturalistic or environmental test setup.

According to [22], the most popular usage of HIL is control validation and verification of ECUs, which are indispensable for vehicles, aeroplanes and robots. Actually HIL simulation test the functions, system integration, and communication of ECUs. Therefore

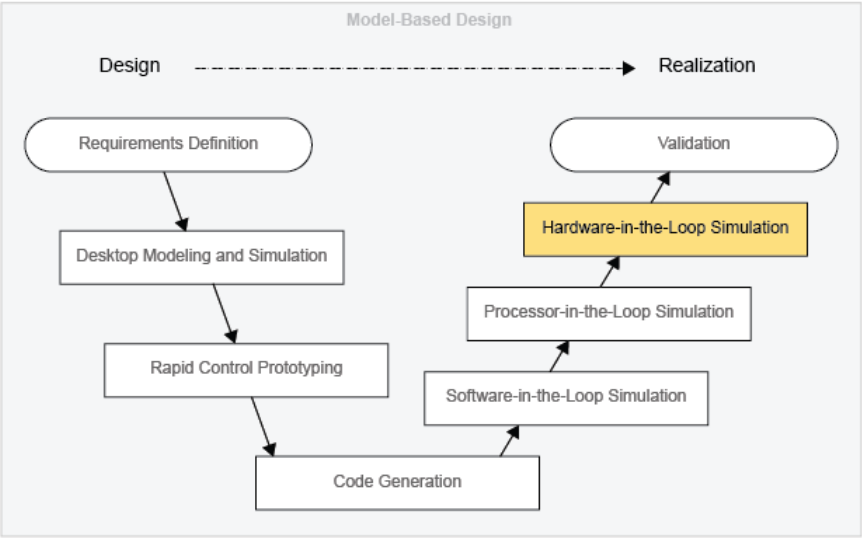

Figure 2.5: Model-Based Systems Engineering (MBSE) design workflow model

the main objective of HIL simulations is to detect errors in ECUs and the situation that produces the error can be reproduced in a completely safe manner, even when critical thresholds are exceeded. In a production system, it is essential to verify the layout drawing (result of the mechanical design), as well as the control program (result of the electrical design), in advance of the real implementation to save the stabilisation time. Conventionally, the mechanical and electrical design activities have been performed sequentially, wasting time. To cope with these problems, in [16] it is proposed a HIL simulation approach for the design and verification of a production system with concurrent procedure of mechanical and electrical design.

2.2.4 Advantages and disadvantages of HIL

Several reasons lead to the development of HIL simulation systems [17]:

- Critical scenarios can be tested without compromising hardware or people. Safety-critical components (such as drive-by-wire) can be tested without danger.
- Test automation is possible.
- HIL tests are cost-effective tests. A scenario is easier to configure in a virtual environment than in a real environment. So, for instance, fewer test drives and test bench experiments are needed, and also fewer vehicle prototypes or vehicle components, resulting in cost saving.

- On one hand, the virtual environment can be easily modified, so a real control (the real ECU) can be tested in different control loops. On the other hand, a real control loop (for example an engine) can be easily tested with different simulated controllers (or control algorithms). Difficult ambient conditions (winter tests, rain, high or low temperatures) can be simulated by simple parameter changes.
- Tests are immediately repeatable, so component failures and associated emergency scenarios can be tested reproducibly.
- The hardware can be tested in parallel in different working environments (for example in a climate chamber).
- If some parts of a system are too complex to model, they can be replaced inside the simulation by the real hardware.
- The embedded real hardware can accelerate the whole simulation.
- The hardware and software of ECU can be tested at any early stage of development. A real engine or a real transmission is not needed.

In [15] even disadvantages of HIL approaches are pointed out:

- HIL schemes are generally not well suited for large-scale simulations with many network entities. Depending on the role of HIL in the simulation, scalability may be a major issue.
- HIL devices may not be readily reconfigurable to consider a variety of different scenarios. Depending on the role of the HIL device, it may constrain the simulation conditions because of its own configuration and/or performance limitations.
- HIL devices will require appropriate interfaces to connect into the simulation environment to exchange data and/or control messages with the simulation engine and the simulated nodes. These interfaces will need to be defined in a clear manner and tested extensively to ensure proper operation of the HIL device(s) in its (their) role in the overall simulation.

2.2.5 Benchmarking

According to [23], in the development of the Electronic Stability Program (ESP) HIL Test Bench, the adjustment of vehicle parameters is a crucial section in the simulation model and it decides the accuracy of the bench to a great extent. The vehicle parameters from

actual measurements are limited, so it is very difficult to obtain a high-accuracy simulation model. The article offers a way to solve this problem: with CANoe from Vector Corp., the experiment data were gathered from actual vehicle and returned to simulation model through CAN telecommunication. This will enable the simulation model to operate in the same mode of actual vehicle. By contrasting the output of response, the vehicle parameters will be adjusted step by step, until the response of simulation model is same as the one of actual vehicle.

2.2.6 Hardware and software tools for HIL

In literature there is a wide treatment of the development of the HIL simulation; in general the papers are organized in two main parts: (1) the presentation and discussion of the model development and (2) the control hardware set-up description [19].

Generally HIL simulation has three main components: simulated components, dedicated hardware systems and real components. According to [15], with reference to the "hardware only" simulation, the designer may choose to program a Field Programmable Gate Array (FPGA) to execute simulation and may find that runtime is greatly reduced for the same set of simulation parameters. An impediment in using this method is the high cost of necessary hardware, which would generally be greater to implement, than the "software only" approach. In general, there is a trade-off between integrating hardware-only simulation, HIL/hybrid simulation, and software-only simulation. In order to determine whether a HIL approach is suitable for a particular simulation, the designer must determine the runtime and/or cost benefit to the simulation design effort.

In [24] a new modeling of field programmable gate array (FPGA) for a permanent magnet synchronous motor (PMSM) speed controller was presented. The whole system is modelled using MATLAB Simulink. The controller is then translated to discrete time and remodelled using System Generator blocks, directly synthesizable into FPGA hardware. This approach allows the modeling of the controller and the controlled system in the same environment, leading to a real time FPGA implementation: the controller was implemented in a low-cost FPGA. But at present time, in the Automation field, Industrial PCs guarantee superior perfomances, since they are equipped with 3rd generation Intel® Core™ processors, see [25], [26], [27].

In the HIL simulation used for the testing actuation system of a 2-DOF Parallel Robot [17], an economical alternative to existing dedicated hardware systems is presented by using the development board FiO Std [28]. HIL simulation includes: target model (the model that runs on FiO Std board), host model (the model that runs on MATLAB/Simulink [29]) and the two servo-motors connected to FiO Std Board.

Concerning the software, there are many off-the-shelf engineering software tools available for HIL testing, like TwinCAT from Beckhoff [26], Automation Studio from B&R [25], ControlDesk Next Generation from dSPACE [30], all of them providing a unique integrated platform which can import simulated processes automatically converted in C source code in MATLAB/Simulink, and use them as a self-contained application module in the control software.

LMS Imagine.Lab Amesim [31] is also a unique integrated platform that provides realistic component and system models, enabling both system engineers and control engineers to start evaluation and validation phases early in the design cycle. But in order to perform a HIL simulation, an LMS Amesim model plant have to be controlled with Simulink or LabVIEW interfaces, and then can be exported to various real-time targets like dSPACE, xPC Target, RTLab, LabViewRT among others.

In [32] it is described the development of a laboratory platform based on the hardware-in-the-loop (HIL) simulation techniques, based on Java language and Eclipse compiler, that are Free Open Source Software (FOSS) tools. The developed module encompasses a three-dimensional digital environment that simulates an industrial plant dynamics, but whose operation is based on hardware components: an external Proportional Integral Derivative controller (PID controller) and an industrial inverter. This HIL platform, totally FOSS-based, allows simulation and analysis of industrial control loops and controllers dynamics belonging to the shop floor operations, and can reinforce the educational laboratory practices for improving engineering education especially that one related to industrial process control, supervision, and optimization.

With reference to hardware PLCs, it is possible to communicate by using OPC (Open Connectivity) technology [33], which provides open connectivity in industrial automation. OPC is an interoperability standard for secure and reliable exchange of data in the industrial automation space and in other industries. It is platform independent, and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation is responsible for the development and maintenance of this standard. In the case of the PLC emulators, the communication can be carried out using COM (Component Object Model) technology [34], which enables software components to communicate.

2.2.7 Model-Based System Engineering and HIL

Mechatronic systems are interdisciplinary, multi-domain complex systems, which involve multiple disciplines, like mechanical, electrical, electronics, and software engineering. The design of these systems inside a multidisciplinary approach allows engineers of different areas to match and interact, developing solutions that offer benefits in global terms and

not just within their scope of responsibility. While in the classic scenario all the mechanical parts had to be ready in order to make then possible the implementation of control, wiring, safety, operator interface, etc., with a systemic approach the mechatronic development process is conceived in a coordinated way in all its parts, and is therefore inherently parallel.

Model-Based Systems Engineering (MBSE) is the formalized application of modeling, beginning at the conceptual design phase and continuing throughout life by supporting system requirements, design, analysis, verification, and validation activities [35]. For the development of mechatronic systems, MBSE allows to apply an integrated approach to the designed systems through a unique core model that works as the repository of the system information.

Unfortunately, current tools do not offer the full end-to-end integration required to support the complete MBSA process.

In the absence of a single unified MBSE tool environment, there is currently a requirement to employ a suite of tools and models of varying complexity.

[9] presents an integrated workflow tool chain to facilitate integration of various subsystems and verification of the overall system performance to meet a set of desired system requirements. The case study example provided has implemented a model-based representation that encompasses real system HIL components to verify the effectiveness and capability of the development platform. This step has demonstrated how MBSE enabled by means of a Model-Driven Architecture (MDA) framework and HIL simulations can be used to support future mechatronic designs by driving the system architecture from a platform independent perspective.

2.3 Virtual Commissioning

In mechatronic machine production, ramp-up phase starts with the completely assembly of a production system and ends with reaching full target quality at a specified cost and a specified output rate. It can be divided into the commissioning and run-up phases. Traditionally during commissioning all activities aim at putting a completely assembled and mechanically reviewed production system into operation. It ends with the production of the first work piece that meets the specifications and the acceptance of the customer. The run-up phase follows the commissioning and transfers the operational production system into stable production conditions in compliance with target cost, demanded quality and output. Analyzing critical points of the ramp-up process of a final assembly, control system malfunctions turn out to be the major source of time delay.

Fig. 2.6 shows that the commissioning phase of a production system accounts for up

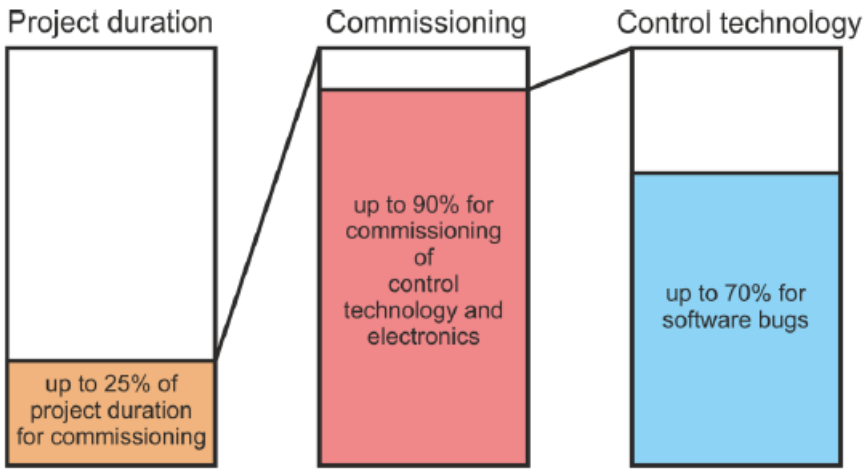

Figure 2.6: Contribution of control software to project delay

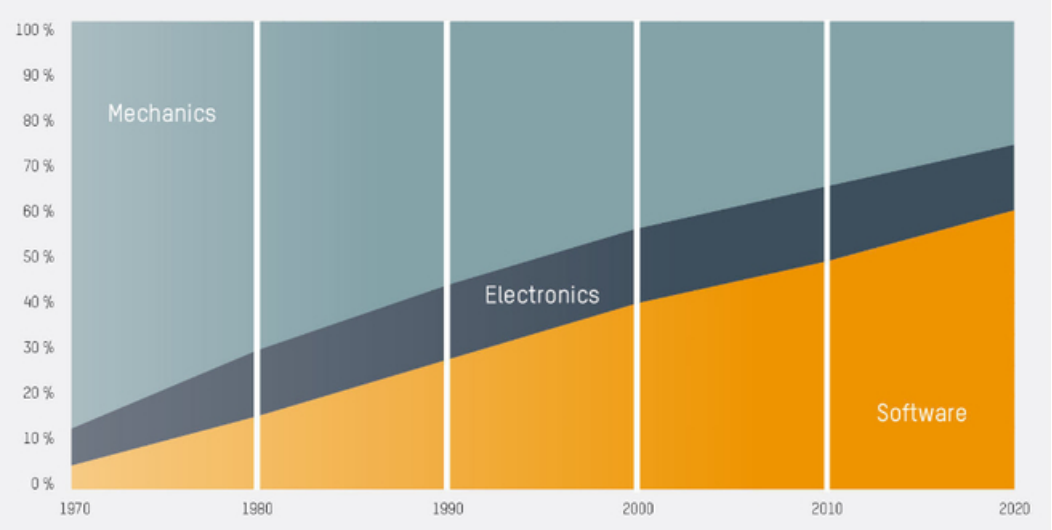

Figure 2.7: Proportion of software development involved when designing new machinery and systems has skyrocketed (source: B&R Automation)

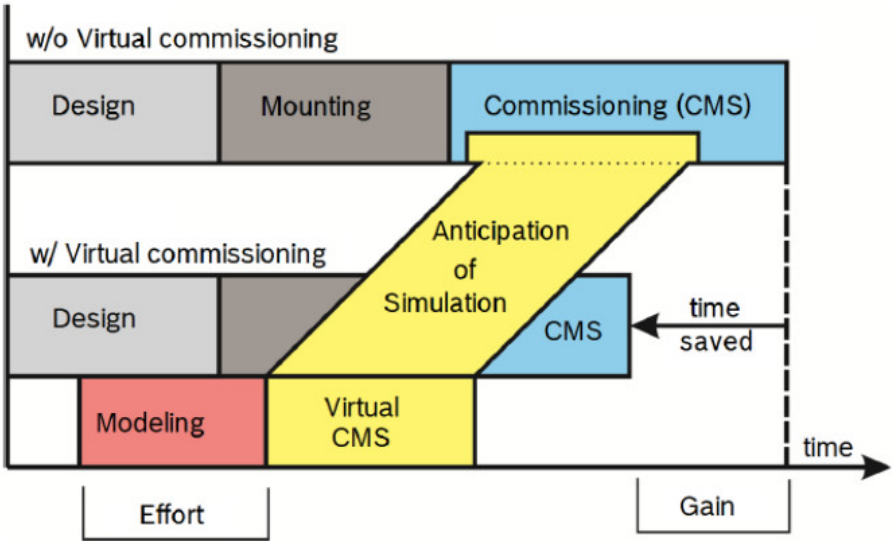

Figure 2.8: Virtual Commissioning procedure

to a quarter of the total project cycle time. A remarkable 90% of the commissioning time is used for delays and activities related for electric and control devices. Again, 70% of this time delay was associated with errors in control software. In other words the corrections of defective control software consumes up to 60% of commissioning time or 15% of time-to-delivery.

Over the last few decades, the proportion of software development involved when designing new machinery and systems has skyrocketed (see Fig. 2.7). Assuming a continuing progression in the contribution of control technology to the functionality of production systems, it can be derived that the delays in commissioning related to software will continue to rise proportionally.

The pressure arising from the complexity of production systems themself and from the growing importance of a short ramp-up phase on the other hand, are addressed by Virtual Commissioning [36]. In this approach, virtual prototypes are used for the commissioning of control software in parallel to the manufacture and assembling of the particular mechatronic system (see Fig. 2.8).

In the research work presented in this thesis, virtual prototypes for control software tests during Virtual Commissioning are built in HIL simulation.

In HIL simulation the production machinery equipment is simulated in real time and connected to the real control hardware.

Costs of the Virtual Commissioning procedure are hardware and software tools, work cost for modeling, and work cost for operating simulations, while benefits are shortening the time-to-market, synergies, and quality improvements.

After the Virtual Commissioning, the next step in a product development workflow is

the physical integration in the Prototype Development milestone.

2.4 Control of Packaging Machines

In [37] the word packaging is defined as "the process of combining products, components, and materials to form a finished package". The packaging production operation is normally accomplished using machinery to fill, weigh, inspect and close or seal a package, as well as to convey, collate and/or coordinate various packaging functions that occur in sequence on the production line.

A packaging machine is typically characterized by highly coordinated movements or functions that are repeated in a short time frame.

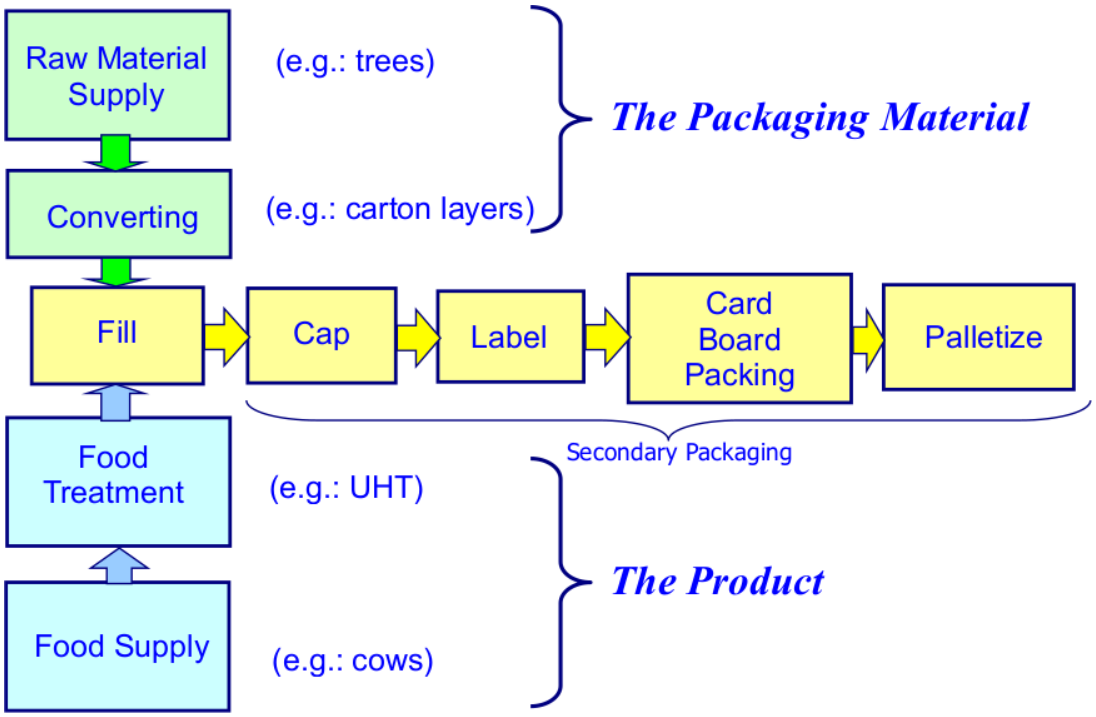

Figure 2.9: Food packaging flowchart

Primary Packaging machines generally shape a packaging material into a container (e.g. a carton package, or an aluminium can) then fills the container with the product (e.g. liquid milk, or carbonated drink) and close it (e.g. seal the carton package, or close the can). In contrast, Secondary Packaging machines generally completes the packaging, sequencing and grouping the containers in patterns, to be inserted into trays or carton boxes and/or shrinked into film, for examples. End-of-Line equipment finally places the boxes onto a pallet (see Fig. 2.9).

A packaging machine treats raw material in successive steps in order to obtain a

finished product, for example it can be transported, sorted, cut, packaged, and boxed in a sequence of operations that must be carefully designed in order to obtain the achievement of a production of good quality. All the above tasks require the manipulation of the product through moving mechanical parts, whose planning and control are very important elements in the process of designing the control system of automatic machines. Generally, the control of the automatic machine can be divided in two hierarchical levels like:

1. A series of logical operations that correspond to various stages of processing of the product. These operations can be represented as a *finite-state automaton*, that is usually implemented using simple combinatorial and sequential logic through a programmable logic controller (PLC).

2. In every state of the automaton, the automatic machine performs the process of the raw product through the coordinated movement of mechanical devices forming a kinematic chain.

More and more the machine needs a mechatronic integration of means, a view that is considering at the same time the four different souls of Mechatronics discipline, i.e. Control Theory, Electronics, Computer Science, and Mechanics. This in order to have higher performance (e.g. the throughput in terms of units per minute) and higher reliability (e.g. in terms of MTBF, Mean Time Between Failure).

A multidiscipline approach is then needed, and a strict collaboration among different groups of designers is mandatory. Mechanical system control is undergoing a revolution in which the primary determinant is becoming the control software. This is enabled by developments in electronics and computer technology.

Mechatronics is simply the application of the latest, cost-effective technology in the areas of computers, electronics, controls, and physical systems to the design process to create more functional and adaptable products. The means to achieve this go through Virtual Simulation and Experimental Verification. The Control Theory view can be described by a control loop formed by a Controller, a computer that implements the control laws for the mechatronic systems, the actuator and sensors that are the interface with the plant, which is the target of the control action. Historically the packaging machines has been driven first by camwheel systems and indexing gears, mechanically linked in order to generate arbitrary motion profile. In modern machine designs, electric servo motors are often used in place of mechanical solutions, providing advantages in terms of solution flexibility and reliability. Indeed a servo system, composed by a motor, feedback system and drive, can follow any position, speed or torque profile and even switch on-the-fly among them. These profiles can also be driven, in turn, by a Master Axis (or Virtual

Master), that is thus synchronizing all the axes or a subset of them. The motion path is repeated every machine cycle.

These arbitrary motion profiles are defined as mathematical formulas that are continuous in time and values, whereas a digital system, such as the servo drive, is discrete in time and values. As a consequence, the Motion Design Process has to be iterative as described in the following algorithm.

1. Definition of coordinated multi-axes motion at load side
2. Definition of mechanical transmission system
3. Mechanical load calculation for each system
4. Definition of coordinated multi-axes motion at motor side. If the solution satisfies the specification go to next step, otherwise, go to step 1.
5. Load simulation of single-axis servo units
6. Control simulation of single-axis servo units
7. Multi-axes simulation
8. Physical test
9. Verification of actual performance

Steps 1 to 4 constitute a highly iterative process that may become very complex for applications with critically high demands. These steps are usually done with help of specific mechatronic tools (e.g. Motion Analyzer from Rockwell Automation). The iterations are needed because of multiple constrains in different fields, that can lead to different optimal solution for the same variable depending on the step we are in.

2.4.1 Nature of Computation for Industrial Automation

In an automation system the controller must take over continuously a physical process, therefore the computation is reactive in nature, that is, the computation is carried out with respect to real world physical signals and events. To achieve this goal, industrial computers must have extensive input-output subsystems that interface with physical signals; moreover, the controller repeats forever the following steps (see Fig. 2.10:

1. Acquisition of sensory signals from the physical process by means of sensing devices (e.g. temperature sensors, position sensors, etc.)

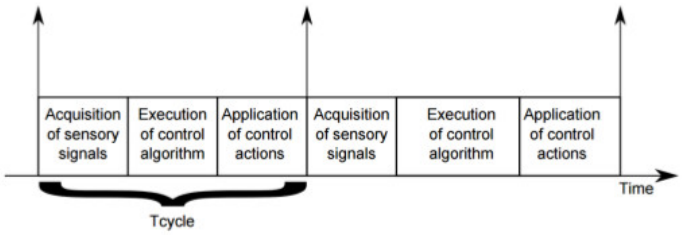

Figure 2.10: Control cycle of a controller for an automation system

2. Execution of the control algorithm, that computes the control actions on the basis of the sensorial signals.

3. Application of the control actions to the physical process through actuation devices (e.g. electrical motors, pneumatic actuators, etc.)

We call T_{cycle} the total duration of control cycle formed by the three above steps: Acquisition, Execution, and Application. The length of the T_{cycle} depends on the speed of the control hardware and the complexity of the control algorithm to be executed, and in general it should be limited after an analysis of the response time required to keep the system under control. The cycle times generally ranges from several milliseconds to few seconds, depending on the mechanical load size and function characteristics.

2.4.2 Real-time systems

A control system behaves correctly if:

- It is logically correct.
- Terminates its execution respecting timing constrains assigned.

Real-time systems fail if timing constraints are not met. As a matter of fact a real-time system must not necessarily be fast, but must respond from external stimuli within a maximum time which is pre-defined.

Real-time systems can be categorized as Hard or Soft. For a Hard real-time, the system fails if a computing deadline is not met. In a Soft real-time, a limited extent of failure in meeting deadlines results in degraded performance of the system, reducing the system's quality of service, but not in a catastrophic failure.

2.4.3 PID: Control theory and Tuning

Servo systems are acting in closed loop systems, that includes:

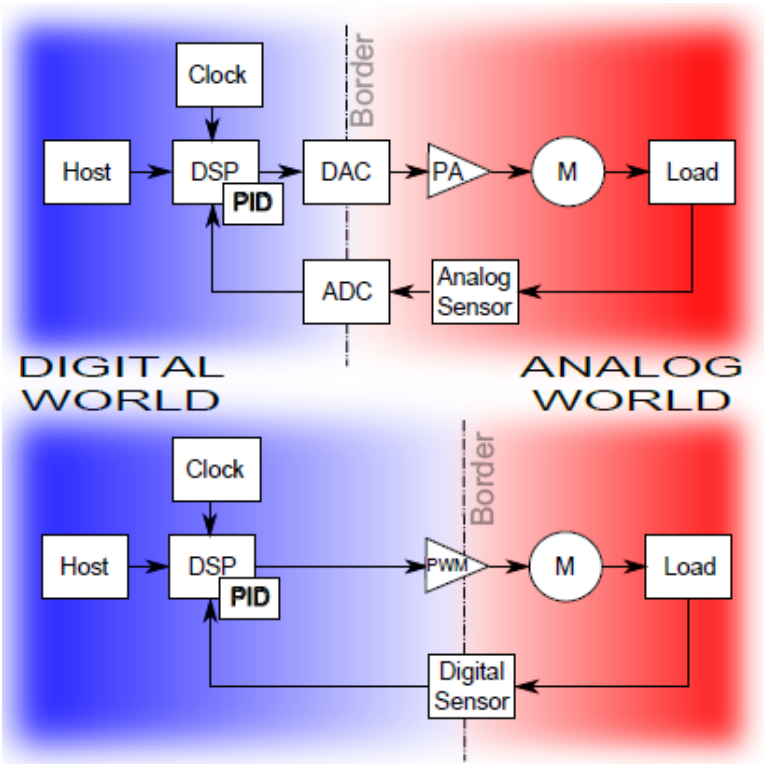

Figure 2.11: The division between the *analog* and the *digital* worlds

- A computation unit or controller (SW and HW, with control algorithms such as PID)
- An Actuation (energy conversion and/or power modulation)
- A target system or plant (that can be a mixture of mechanical, fluid, thermal, chemicals, electrical aspects)
- A feedback or sensor (energy conversion and/or signal processing)

All that interacts with the machine or process environment, and digitally communicates with the operator through a machine interface (equipped, for example, with a touch screen HMI), and with all other machine components through fieldbus and/or wireless communication.

The controller is often using a PID algorithm, that nowadays is done in a digital form as shown in Fig. 2.11. The border between digital and analog worlds has been moving more and more from left to right in this scheme.

The PID, or PIDs, for motor control can be placed on drive or on central controller, depending on motion control architecture and on automation supplier. This choice have an

impact on field bus effective bandwidth since the amount of information to be exchanged is affected.

The PID, as the word says, includes: A Proportional Action P that, if increased, can:

- Make the system more reactive
- Reduce the effect of noise: for high G values, the transfer function G/(1+GH) goes toward 1/H that is independent from G or eventually lead to overshoot or even instability if increased too much

Integral Action that, if increased, can:

- Reduce steady state error
- Eventually lead to low frequency oscillations or even instability if increased too much

Derivative Action that, if increased, can:

- Act as a damper
- Work against high slopes in the controller action
- Eventually lead to high frequency oscillations or even instability if increased too much

The PID is applied on an (e.g. position) error that is created through comparison of set point and feedback data. The feedback sampling is decided examining the application real time needs, and considering Shannon Theorem. In particular strict real time applications requires tight sampling, that in turn acts as an enabler for high gains, high bandwidth and high performances. The Shannon Theorem tells us that once we know the frequency content of interest of the target system, then we have to set the sampling frequency at more than double that amount. Generally the choice is for five or ten times more, for avoiding complex modeling with Z-Transforms.

Stability of the closed loop system can be addressed with Nyquist Criterion or Root Locus. This latter one is giving added information such as how reactive the system is (poles on the right of the complex plane), if it oscillates (poles out of the real axis) and at which frequency (distance from the real axis).

PID tuning

PID gains, system bandwidth, and system performances are linked.

Let's assume that the set point is varying with a step.

The position (or lag) error will suddenly increase, given that the system has a finite energy (being of second order or above) to respond to the immediate set point change. Then the system will begin to move with a certain inertia and lag error will finally create a negative peak when the set point is reached, to decelerate the inertia. After this peak, the lag error will somehow go back to zero once the motor moves to close the gap.

Let's focus on P action of PID, and neglect the I and D actions. P action indeed is directly acting on system bandwidth, as shown in Fig. 2.12. At the output of the P (PID) action there will be the same peaky waveform enhanced by a factor proportional to the P gain value. This, in a simplified scheme, with a parallel PID, is the current to the motor, that is its acceleration. The motor will then accelerate to reach the set point (first peak, positive) and finally decelerates when the set point is reached (second peak, negative).

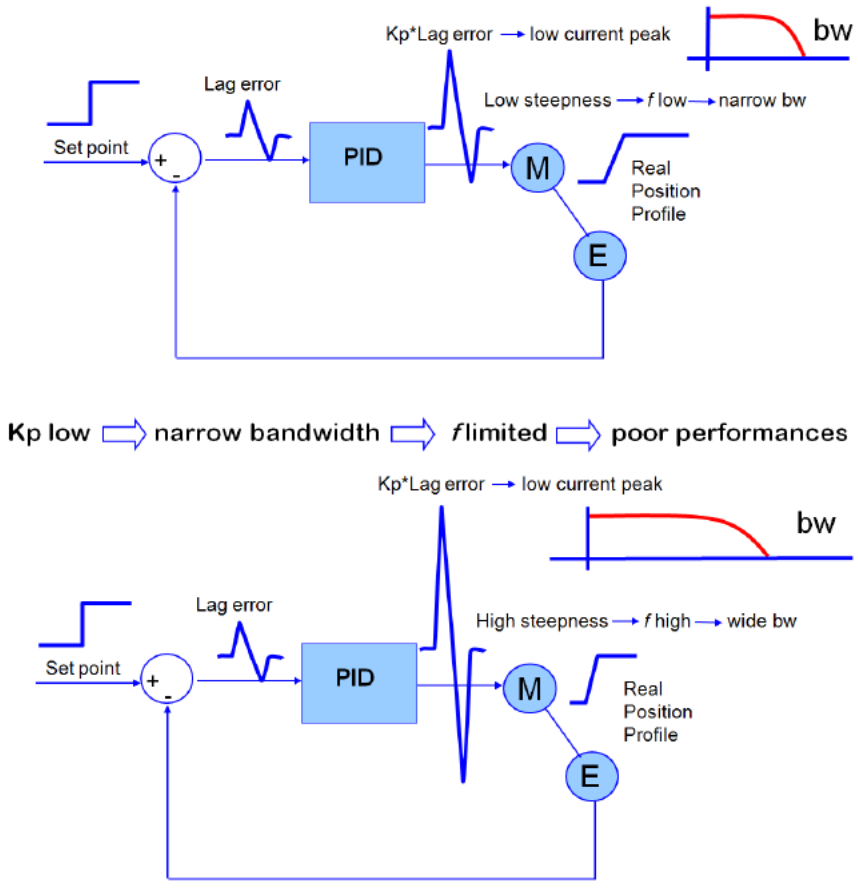

Figure 2.12: PID Proportional gain ("P" term Tuning)

Now, if we increase the P gain, the the PID output peaks will be higher, than in turn will create more motor acceleration, quicker load response, i.e. higher bandwidth, i.e. higher system performances.

The system depicted above is a simplified one. In practice there will be other blocks in series to the PID, such as a Notch Filter.

Several approaches exist for PID tuning.

The most used approach in industry is the experimental approach.

The Step Response PID Tuning is an experimental method in which the basic approach is:

- Initialize the I term to zero, and set the D term to a small non-zero value.
- Use a step-response as reference:
  - Increase P from zero until the system substantially overshoots.
  - Then increase D until the oscillation is "critically damped".
  - Continue this iterative process until we get the right system performances or meet instability, in which case we step back for keeping a safety margin (typically at least 20%)

Although very easy to use, this method has the problem that increasing D will cause the optimum value of P to change, which in turn changes the optimum value of D, and so on. This requires a number of iterations to get to stable values. In general terms, this is because the D term of a PID operates at the highest frequency zone, the P term at a middle zone, and the I term at the lowest frequency zone [38].

In the research activities described in this thesis, P position gains and P velocity gains have been applied to all control problems with success.

2.5 Packaging line design

This section will use the terms machine builder, or vendor, to identify the company responsible for the design, fabrication and assembly of the completed packaging machine. The producer, or the customer, is conversely the company which orders the packaging machine procurement.

In virtually all cases, the machine builder will purchase certain standard machine components, such as motors, controls, bearings, and more, from third parties.

In Fig. 2.13 is outlined an example of plant of a food producer.

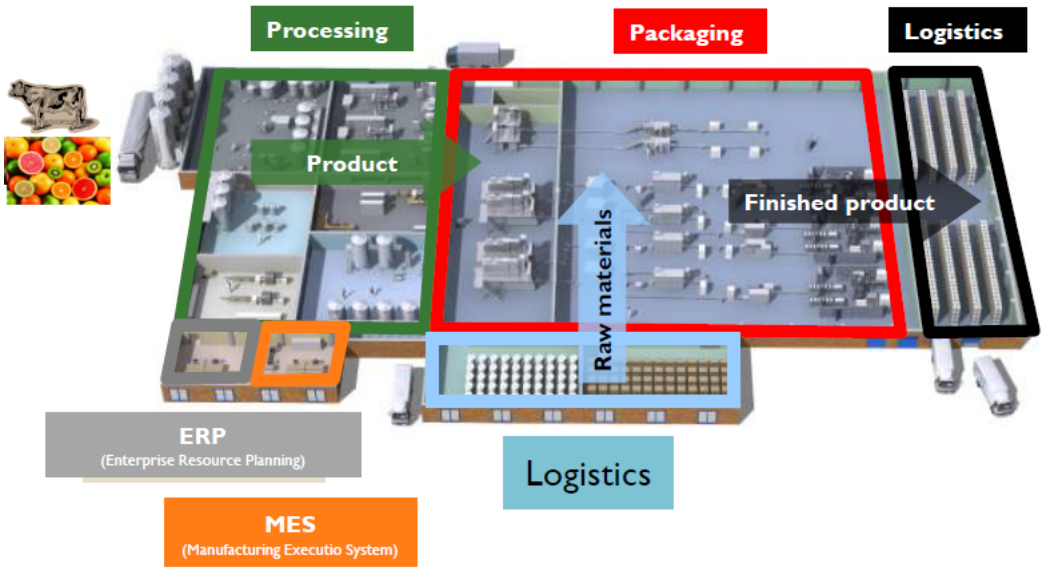

Figure 2.13: An example of plant of a food producer

The typical packaging line consists of number of machines, often supplied by different machine builders. The key distinction between a packaging line and a collection of packaging machinery is, on the packaging line, the machines work smoothly together to efficiently produce a high quality product. The producer will sometimes have the luxury of being able to design the facility around the packaging line. In many cases, it will be needed to fit the line into an existing facility, which will require tradeoff between line and facility designs. Some packaging equipment will be standard, off-the-shelf design. Most require some modifications. Some may be custom designed and built from the ground up. Existing equipment may need to be reused even if it is not what would be selected in designing a line from scratch. In other cases, various constraints may require the purchase of used equipment. All these factors add to the challenge of designing, purchasing and commissioning a packaging line. There is no "one size fits all" approach to packaging line design, every line will be different with different requirements and constraints.

In the following, a general design framework for producer will be briefly summarized. The framework divides the line design process into five sequential steps:

1. Determine requirements
2. Develop a plan
3. Procure equipment
4. Install and commission the line

5. Begin production

Determine requirements The first step in designing a packaging line must always be to gather information. The more information obtained before starting the project, the easier the designer team job will be as the project progresses. More information at the beginning helps guard against (almost always expensive) surprises later. More information, widely shared, also helps guard against disappointment due to differing expectations of what the line is to accomplish and how.

Information is an iterative process. At the beginning of the project, information will probably not be completed and is often subject to change. Major areas will include the package and its components, the product, the room where the line will be placed, utilities available, legal and regulatory issues, workforce, transport and more.

Designing a line for even the simplest package can be complex. It is important to get as much information as possible, including samples, as early in the design process as possible. The package designer and the packaging line design engineers should consult early and often.

The information required can be organized into three main areas: budget, package and block layout.

- Budget - Until the project is not well defined, it will be hard to develop anything better than a ballpark budget. This may be off by as much as 50% to 100%. This ballpark budget is still useful because it may prevent time and effort being wasted on a project that is *prima facie* unfeasible. As the project develops, the budget must be continuously refined and updated. There are two main approaches to budgeting, with many variations in between. One, a forward approach to budgeting, gives the design team free reign to develop the optimal design with costs calculated based on the design. Two, a backward approach to budgeting gives the design team a budget amount and asks them to design the optimal line within that constraint. In many cases, some middle ground will be used and approval of the funds and line design will be based on negotiation of costs and benefits along with capital availability. In addition, a time budget should be developed as early as possible. This, too, can be forward, with the projected finish date calculated based on design requirements, or backward, with a fixed completion date and the design managed to comply with the date.

- Package - To obtain crucial information about the package, designers will need to verify these critical details:

- Component specifications: detailed package specifications and drawings must be supplied to the design team. 3D prototypes can give insights which two-dimensional drawings may not be able to.
- Bill of Materials (BOM): a BOM is a complete list of all components and materials required for the final package. This will include primary, secondary and tertiary packaging.
- Product: complete details, including samples, of the product are necessary as early as possible. Product information must include how the product will be supplied to the line, such as pumped from a remote reservoir, small hand-loaded totes or pails, or large machine-loaded bins. Some products may have special handling requirements. They may need to be packaged hot or cold. Fragile or delicate products may require gentle handling. Most caps may be handled in a standard rotary sorter with no ill effects. But a cosmetic product, such a perfume, may have a closure with a highly polished finish that would be damaged with normal handling. The line designer will need to provide a suitable alternative. Other products such as pharmaceutical, medical devices or foods may require a "clean" production environment. "Clean" in this case refers to a specific design philosophy requiring special machine design, special air filtration and flow, room design, construction materials, and personnel operations. Some products may be flammable or explosive, or present chemical or biological hazards, requiring specially configured equipment.
- Facility: the packaging line design team may not have direct responsibility for facility design issues, but must be aware of them as they will affect the success of the line design. The line design team and facility designer must work closely together to determine what is optimal, what is feasible and how to reconcile the two. Physical space and configuration may be the most obvious criteria but are hardly the only ones. If design team is required to place the line in an existing facility, they must know all details of the available space. The first is room dimensions. Not just size in total square meters, but how that space is laid out.
- Utilities: designers must know what utilities are available. The first utility that most designers think of is electricity. Designers purchasing a machine from another country for use in the EU or purchasing a EU built machine for use in another country, must be sure that the machine is built to the appropriate electrical standards for the country where it will be used.
- Workforce: a line can be highly or even totally automated. A highly automated

line requires a relatively small workforce, as the machines will do most of the work. The automated line will be more complex requiring a more sophisticated workforce.

  - Capacity requirements: required capacity is usually the key parameter in any line design. For the purpose of this section we will consider capacity as the quantity of finished products per year. Production quantity is important, but equally important is when the capacity is required. Some product have relatively stable demand over the course of the year. Other products have fluctuating demand. Designers need to know what production strategy the company wishes to pursue to properly size the line. Production variety will also affect capacity. The production fragmentation has been increasing for the last several decades and shows no signs of letting up. The ever-increasing product variety results in lines that need frequent changeovers. The amount of time lost will depend partly on machine design, partly on package design and partly on the product. Even relatively small improvements in changeover times can have large cumulative impacts. Saving 10 minutes per day will, over the course of a year, free up more than 40 additional hours, or one week, of additional available production time.
  - External requirements: many external requirements impose conditions on the line design. Governmental regulation is perhaps the major one. E.g. pharmaceutical lines will be regulated by the Food and Drug Administration (FDA).
  - Company policies
  - Industry practices

- Block layout - Early in the design process, a block layout or flow chart should be developed. This is a simplified line layout with each machine represented by a box or a block. The blocks may or may not be to scale. They may or may not be arranged as the line will be (straight vs. U). The purpose of the block layout is to show the major independent pieces of equipment required and their sequential relationship. Connecting lines are used between boxes to indicate conveyors.

The above are some of the issues that must be addressed in the early stages of the packaging line design. Not all will apply in all cases.

All parties involved in the project must agree on the various parameters and criteria. These parties obviously include the package, facility and packaging line designers. Other departments such as finance, purchasing, manufacturing/packaging operations, quality, validation and others must be involved as appropriate.

Develop a plan The timeline, or time budget or schedule can be as important, or more so, as the monetary budget. In case of a new product, time to market may be of the essence. New consumer products that do not get to the market soon enough, may find themselves beaten out by a more nimble competitor.

- Develop an installation and integration plan: the design team will need to determine who will install the line once it arrives at the plant. Most lines consist of multiple machines from multiple vendors, often in multiple countries. Simple installation of the machines is not enough. They need to be integrated together as a single packaging line. This includes interconnection of controls and sensors, utility runs, conveyor transfers and more.

- Write specifications: after the above questions have been answered, it is time for the design team to begin writing the specifications for each machine in the line. Specifications are the communication between buyer and builder about what is wanted and what is expected. Once accepted by the machine builder or vendor, they constitute a legally binding commitment as to what is to be provided, how and when. As important as the legal aspect is that it avoids any misunderstanding.

- Evaluate whether to consider used equipment: line designers will likely go into a project with new machinery in mind, but used machinery may be a viable alternative for some projects. Consider that "used" equipment can include never installed equipment still in original crating as well as producer owned equipment.

Procure equipment Once the specification has been written, identify potential vendors. This can be quite a task; in some categories there may be dozens of vendors who are, at least theoretically, capable of supply a machine that will work.

Consider capability: some vendors have developed specialized expertise in particular industries or product types. Financial capabilities is critical as well. In the worst case, a financially weak vendor may go out of business with the machine half built. This can result in the lost of the funds paid. Worse, it puts the project back to square one with the need to find another vendor, go through the purchasing process and then wait for delivery. A financially weak vendor may also have trouble paying their vendors. This can result in delays of materials and, ultimately, delays in delivery of the machine. For checking the vendor's reputation and reliability, ask the vendor for the names and contact information of several users running similar machine and similar product under similar conditions to those under consideration. Then talk to those users and find out their experiences. Vendor staffing, both quality and quantity, is important.

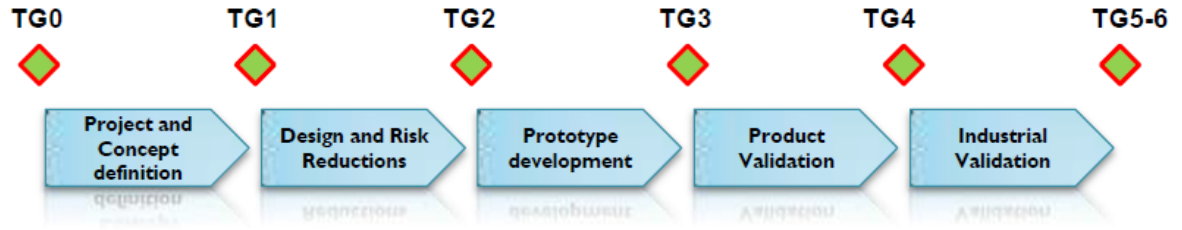

Figure 2.14: A product development process

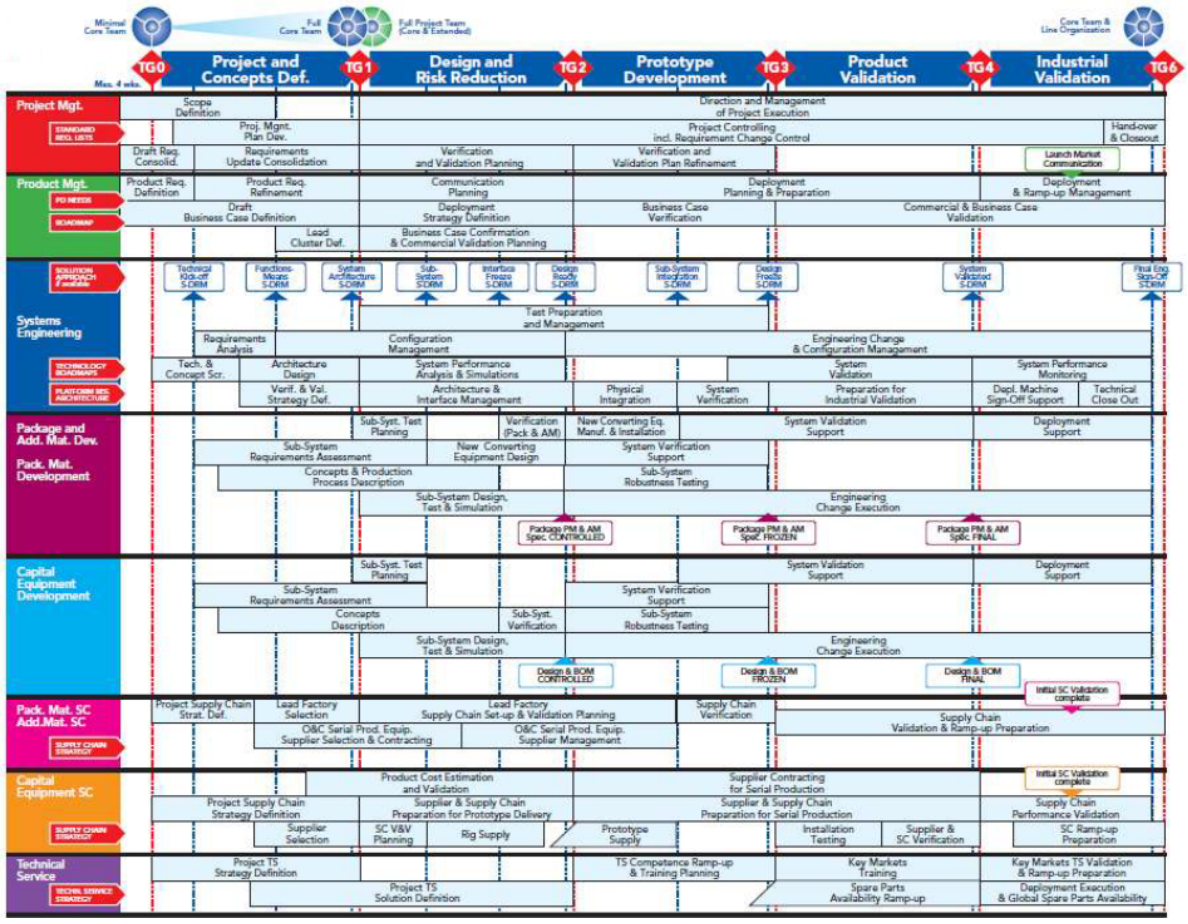

Figure 2.15: Workflow of a product development process

From the machine builder's point of view, the objective is to transform customer's needs into a commercially available product.

In Fig. 2.14 is outlined an example of product development process. As it can be seen in Fig. 2.15, the workflow of a product development is challenging and involves several teams in the organization of all project activities [39].

In a product creation, besides project management, workflow, and governance, a crucial element is System engineering. Systems Engineering is a requirements-driven ap-

proach to technical management in product and technology development projects. Its primary aims are to:

- deliver the product right the first time
- meet the customer expectations with the right product

Since is proven that most of "defects" are generated in the early project phase either by wrong, unclear of missing requirements, a good requirement engineering approach is key to avoid rework that will heavily impact project cost and timeline during the project life cycle. This approach starts with the definition of customer needs, and identification of product functionalities. The intended product validation is also done very early in the life cycle (V diagram). The success of the Systems Engineering approach lies in reducing the complexity inherent in developing systems. It does this by organizing the development of the full system into a set of smaller, confined and more manageable sub-systems.

Install and commission the line Upon acceptance at the factory, the machine may be shipped. Shipped must be done properly to assure that there is no damage in transit. After having spot the machine in the point of installation, it should be aligned, levelled and fastened to the floor as appropriate. Then commissioning is performed, including utility connection initial startup and testing in place.

Begin production At this point, the line should be ready to run its first production batches. Ideally, one should be able to flip a switch and have the line running at normal productivity efficiency. In reality, production startup will always be a learning curve. Part of it may be due to things that were missed in the line design process up until this moment of truth. Part of it will be the newness, to this line anyway, of operators and technicians.

In conclusion, successful designers strive for creative solutions to the problem of combining the variety of components, materials and product into a high-quality finished package. Attention to detail beginning at the earliest stages of the project can have impact that lasts years and decades after the line is installed. Even a 1% gain in line productivity or efficiency can save the company millions of euros over the life of the line [37].

2.6 Discussion

Unanticipated emergent properties and system behaviour that can occur later on in a product's life-cycle have the potential for expensive rework: there are many examples of

serious unanticipated emergent behaviour that has appeared directly once a product is in service. This implies huge financial and reputational cost to the companies involved. HIL plays a key role in providing the means to test products from very different fields at an early stage in product's life-cycle.

According to [8], in the future both real-time computing and plant modeling will close the gap of complete bench top simulation and may make the HIL simulation an absolute replacement for a real vehicle for control system validation and verification. Furthermore, future advances in the HIL test system for vehicle will lead also to full vehicle integration control with active differentials, anti-block braking, electronic stability and traction control. In addition, the virtual reality technology will be used for HIL test system.

Beyond Virtual Reality and Augmented Reality, a promising software for HIL aims is Massive (Multiple Agent Simulation System in Virtual Environment), a software package with a flexible Artificial Intelligence–authoring environment for modeling the idiosyncrasies of complex, real–life behaviors into agents who use visual and auditory cues. One key differentiating characteristic of Massive is that it employs an Artificial Life approach to AI. Artificial Life technology draws from the processes of nature rather than traditional simulation methodologies, providing for more inherently natural behavior than those seen in existing engineering solutions. [40].

Chapter 3

Software architecture

3.1 Introduction

Nowadays the creation of the model of a mechatronic machine no longer requires complex mathematical functions to be written. The work is facilitated by the presence of extensive function block libraries, suitably configurable, which can be composed to create a machine model, however complex, within the physical system, including electric drives. The code automatically generated by the combination of the different blocks can be analyzed and refined later, if necessary. Today it can be offered in real-time what in the past was obtained by performing tedious hand calculations and inverse kinematics functions with approximations. This results in a greater accuracy, less possibilities of errors, and a code more robust and faster to execute. The code related to a machine model includes the control algorithm as well as the plant model. It is created with dedicated tools, and can be integrated directly into the HIL platform, where it runs in real-time directly on the physical hardware of control.

The HIL simulation is extremely efficient and reduces the modeling, coding, and implementation time, because every aspect of the project and of the system optimization can be easily addressed on the development personal computer.

The mechatronic development brings huge advantages in terms of time and cost of implementation, but it must be supported by a complete design environment, integrated and opened in order to exploit the potential of the MBSE approach.

This chapter describes a model-based representation that encompasses HIL components to verify the effectiveness and capability of the development process. The identification of a suitable model-based systems design methodology and relative toolset is presented and includes HIL both for system verification, and for validation of a novel HIL scheme. Firstly the commercial softwares employed in the present research work are listed. Then a formal method is proposed in order to define the activities to be performed

for the design development of a mechatronic machine. Finally it is presented the software architecture implemented, with description of the employed software development technologies and of the activities to be performed to create a workflow and verification of overall system performance, in order to meet the desired system requirements.

3.1.1 Outline

The outline of this chapter is as follows: Section 3.2 introduces the most important features of the tools employed in this research work, while Section 3.3 describes the development process that is behind the software architecture of a mechatronic machine project. In Section 3.4 the proposed system architecture and its software implementation is presented.

3.2 Software tools

3.2.1 No Magic MagicDraw SysML PlugIn

The Systems Modeling Language (SysML) is a general-purpose graphical modeling language that supports the analysis, specification, design verification, and validation of complex systems such as hardware, software, data, personnel, procedures, or facilities. For all its benefits, people from academia and industry world decided to utilize it as a tool for the application of the MBSE approach [41]. SysML is a customization of the Unified Modeling Language (UML) for system enginnering applications, supports hierarchical modeling and allows the representation of these system views: structural, functional, behavioural, requirements and constraints. It is intended to model systems from a broad range of industry domains such as aerospace, automotive, health care, and others.

Nowadays SysML enables MBSE, however SysML is neither an architecture framework, nor a method. Unfortunately current tools do not offer the full end-to-end integration required to support the complete MBSE process. In the absence of a single unified MBSE tool environment, there is currently a requirement to employ a suite of tools and model of varying complexity [9].

This opens discussion of:

- how to structure the model,
- what views to build,
- which artifacts to deliver and in what sequence.

Every company deals with these issues differently and it results in the loss of capability to inter-exchange, loss of capability to communicate with other teams, overhead in tool

customization, and specific training need. Moreover, the models become impossible to integrate and reuse [42].

Since all the model integration possibilities are still difficult to implement, time-consuming and not robust enough [43], the purpose of using SysML in this research work has been to give a standard description of the project development. In order to trace where the information can be found, hyperlinks will connect the SysML models and domain specific models.

The chosen SysML software is No Magic MagicDraw SysML PlugIn. Before selecting No Magic MagicDraw SysML, also IBM Rational Rhapsody and IBM Rational Harmony for Systems Engineering have been tested. Eventually it was decided that No Magic MagicDraw SysML was better suited to our needs. SysML tool selection has been based on an evaluation against criteria that include hands-on use of the tool in the expected environment, usability, and review of vendor information.

3.2.2 MATLAB/Simulink

Simulink is a block diagram environment for multidomain simulation and Model-Based Design developed and sold by The MathWorks, Inc.

Simulink supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems and is in widespread use in research and development since many years.

One of the strengths of Simulink that should be emphasized in conjunction with automation technology is the design of controllers at simulation level. The functions of the higher-level software environment MATLAB (with appropriate toolboxes) enable the easy design of controllers using calculation methods based on closed loop control theory. Controllers designed in this way can subsequently be simulated and validated together with a model of the controlled system in Simulink. But also entire production plants can be modelled and tested in simulation. This is advantageous in various ways in the process of developing a machine. For example, no real machine needs to be available for initial tests of the control software. Moreover, the danger of damage to the machine through initial commissioning with flawed control software is reduced.

In the Simulink family products, useful specific libraries for the physical modeling of mechanical, electrical, and other physical domains components are Simscape, SimDriveline, SimElectronics, and SimMechanics.

SimMechanics provides a multibody simulation environment for 3D mechanical systems, such as robots, vehicle suspensions, construction equipment, and aircraft landing gear. You model the multibody system using blocks representing bodies, joints, con-

straints, and force elements, and then SimMechanics formulates and solves the equations of motion for the complete mechanical system. Models from CAD systems, including mass, inertia, joint, constraint, and 3D geometry, can be imported into SimMechanics. An automatically generated 3D animation lets you visualize the system dynamics.

Stateflow is an environment for modeling and simulating combinatorial and sequential decision logic based on state machines and flow charts [29].

All of above MATLAB packages have been employed for modeling the systems simulated in the present research work, as reported in Chapter 4.

To use control programs and controllers designed in Matlab/Simulink with the real machine after successful tests in simulation, the developed algorithms can be programmed manually in real-time-capable languages like C++ or PLC code. Simpler and significantly less prone to error is the automatic conversion of the already implemented algorithms into real-time-capable program modules.

3.2.3 Beckhoff TwinCAT

Overview

TwinCAT (The windows Control Automation Technology) is a PC-based control software and is produced by Beckhoff Automation GmbH & Co.KG.

TwinCAT transforms every compatible PC into a real time control with multi-PLC, NC axis control, a programming environment and a control station. By means of its features, TwinCAT substitutes PLC and NC controllers as well as control stations. In order to realize control applications, the user can access different programming languages: the classic PLC programming languages of the IEC 61131-3, or the high level languages C and C++, as well as MATLAB/Simulink.

TwinCAT is classified into eXtended Automation Engineering (XAE) and in eXtended Automation Runtime (XAR).

One of the main approaches of TwinCAT is to simplify the software engineering. Instead of developing own standalone-tools it is obviously worthwhile to integrate into common and existing software development environments. For TwinCAT this development environment is the Microsoft Visual Studio. By integrating TwinCAT as an extension into the Visual Studio, the user is provided with an expandable and future-proof platform.

If you install a full version of the Microsoft Visual Studio, the whole functional range including C, C++ and also MATLAB/Simulink is available, as illustrated in Fig. 3.1.

The ADS protocol (ADS: Automation Device Specification) is a transport layer within the Beckhoff TwinCAT system. It was developed for data exchange between the different software modules, for instance the communication between the NC and the PLC. This

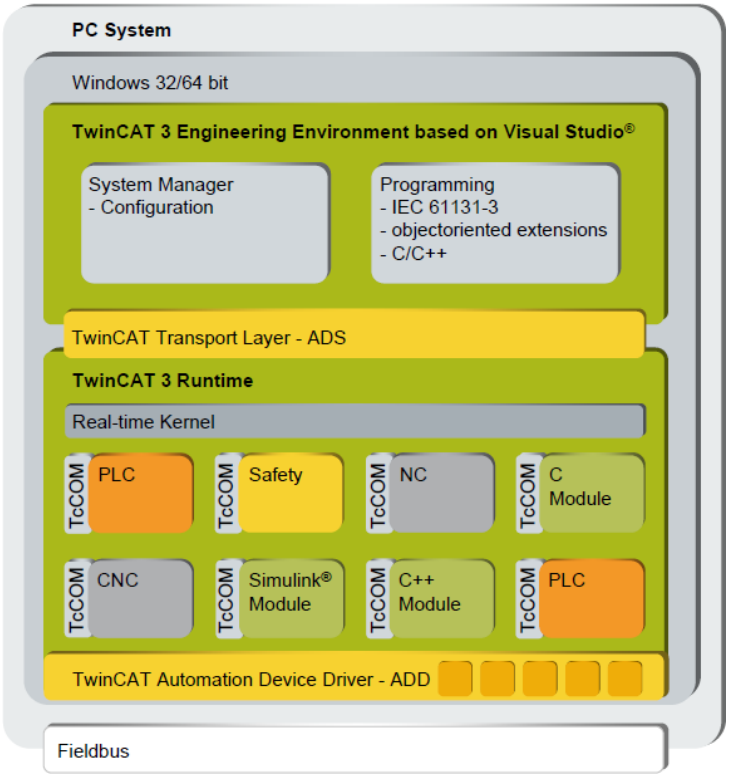

Figure 3.1: TwinCAT eXtended Automation (XA) architecture

protocol offers the freedom of using other tools to communicate with any point in TwinCAT. If it is necessary to communicate with another PC or device, the ADS protocol is used on top of TCP/IP. This means that in a networked system, all the data is accessible from any desired point.

The TwinCAT Runtime offers a real-time environment where TwinCAT modules can be loaded, executed or administrated. The individual modules must not be created with the same Compiler and therewith can be programmed independent and by different manufacturers or developers. Furthermore it is not important if the modules are PLC, NC, CNC or from C-Code generated modules. The generated modules are called cyclically from the Tasks. Several Tasks can run on one control PC.

Another key feature of the TwinCAT Runtime is the supporting for multi-core CPUs. This is fundamental for a software operating on a Personal Computer.

TwinCAT has a modular architecture. All the real-time components are encapsulated in modules which are managed by the run-time system.

TwinCAT uses the concepts from “Component Object Model (COM)” to define the characteristics and behavior of the modules. TwinCAT COM (TcCOM) is the adaptation of COM to the automation technology that allows modules implemented in different

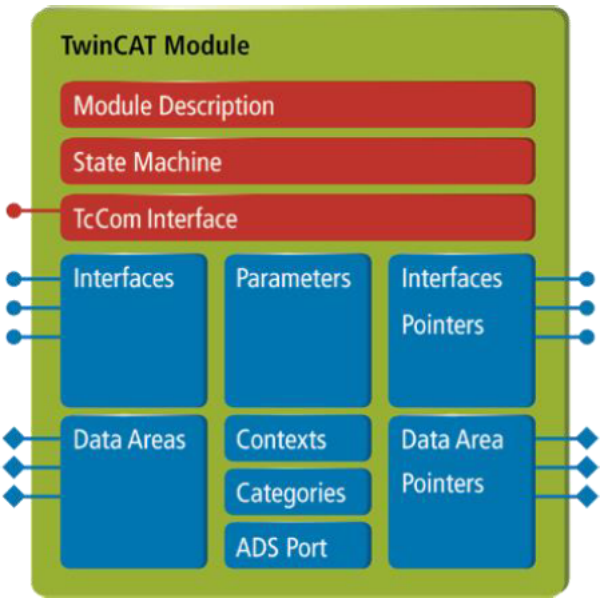

Figure 3.2: Internal structure of a TwinCAT module

languages to interact seamlessly in the real-time context.

Each TwinCAT module has a set of mandatory and optional attributes. The mandatory attributes of each TwinCAT module are: description, state machine and a generic interface (ITComObject). The ITComObject interface is used to access basic information and status of the module like name, object ID, parameters and state. See Fig. 3.2.

The module state machine, as showed in Fig. 3.3, describes the general state of the module.

It controls the inizialization, parameterization, and the creation of the connection to the other modules.

There are two types of files to describe the modules:

- Class description files, *.tmc
- Instance description files, *.tmi

The class description contains the description of the module and its interface together with the general information of the module: vendor information, class ID, etc. The instance description file contains the concrete settings of the module like parameters, interface pointers, etc.

In this research work it has been employed TwinCAT version 3.

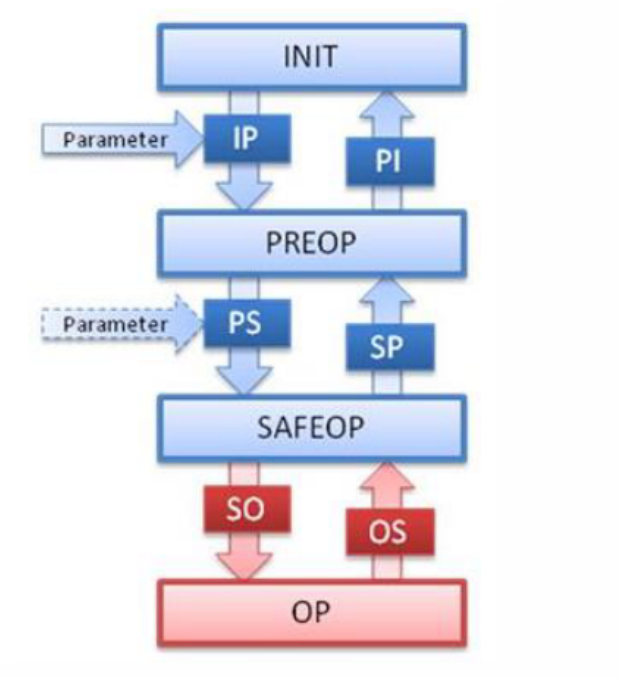

Figure 3.3: TwinCAT module state machine

TE1400 TwinCAT Target for Matlab/Simulink

The TE1400 TwinCAT Target for MATLAB/Simulink enables the user to generate real time capable modules, which can be executed on the TwinCAT runtime. These modules can be instantiated multiple times, parameterized and debugged in the TwinCAT engineering environment.

The "Simulink Coder" (formerly "Real-Time Workshop") generates real-time-capable C or C++ code from Block diagrams implemented in Simulink. The TwinCAT Target for MATLAB/Simulink uses the Simulink Coder to create a TcCOM module (a callable real-time-capable TwinCAT module) with the input and output behaviour according to the source Simulink Model. Generated modules can be instantiated in TwinCAT projects, where those can be parameterized using the TwinCAT Engineering Environment (XAE), if parameter changes are necessary. After starting the TwinCAT runtime, the module is executed in real-time and can thus be integrated into a real machine controller.

A possible area of application of the TwinCAT Target for Simulink is HIL, in which a Simulink model of a machine is translated to a TwinCAT module in order to test a PLC program in real-time, even before the real machine is connected. This application, both improving the design and development of machine model and the control logic test, is called Virtual Commissioning.

Individual parts of a complex system can progressively be encapsulated in separate

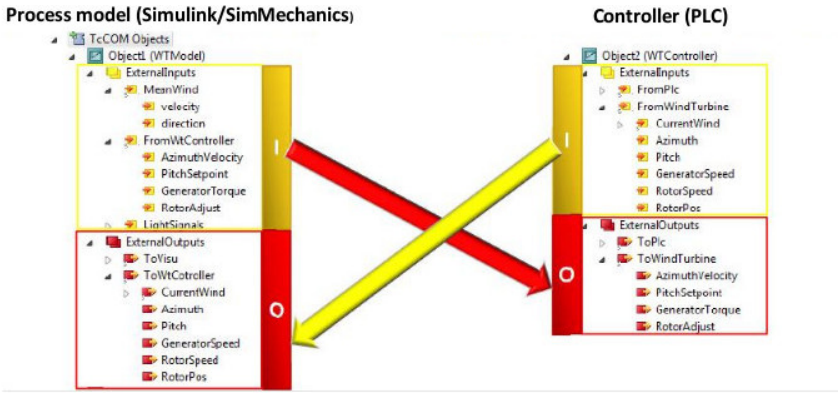

Figure 3.4: Interaction between Process model and Controller

Simulink models, which are joined together in an overall model by means of Model Referencing. By encapsulating the sub-systems, TwinCAT modules can be generated from each of it. These modules are used for the real-time simulation of the sub-systems in TwinCAT. The real system is put into operation successively, while individual system components are still integrated as simulation models.

In Fig. 3.4 is represented the interaction, in a TwinCAT simulation, between a Process model and the Controller in real-time, with the mapping [26].

Motivation

Beckhoff TwinCAT is the software used to implement the brand new HIL scheme proposed in this thesis for the following reasons:

- PC-based automation technology
- one programming environment, one project file, one debug environment
- usage of C++ for real-time code
- link to MATLAB/Simulink
- modular control software

3.3 Development process

According to [44], the solution to reduce the cost development and validation of simulation models is simplifying the model development process through modularization, and the creation of re-usable simulation-model code and data. Such neutral simulation-model

formats could be used to develop shareable models by individual companies, simulation vendors, equipment and resource manufacturers, consultants, and service providers. Therefore instead of coding models from scratch and building custom data translators to import required data, a better solution would be to simplify the process through modularization, i.e. the creation of re-usable simulation model building blocks. By the use of standardization, periodic or labour-intensive tasks can be simplified and become more efficient. This leads to a higher overall efficiency in simulation projects, which would be constructed by assembling or configuring modular building blocks [45]. Indeed simulation object libraries are frequently used in a great variety of sectors, from automotive industry, through shipyard sector, to sawmill sector [46].

Modularity is one of the Industrie 4.0 Design Principles. Implementing modular systems would lead to flexible adaptation to changing requirements and easy adjustment in case of seasonal fluctuations or changed product characteristics. In the Smart Factory plant, new modules will be added using the Plug&Play principle [47].

In this research work the modular approach is applied to the design and testing of a packaging machine.

The testing is performed through modular HIL simulations, employing a brand new HIL scheme. Therefore the packaging machine design testing has also the aim to validate this cutting-edge HIL environment.

Following the MBSE approach, the applied methodology is developed as follows:

1. Requirements collection
2. Modular System Design
3. Implementation
4. Testing

3.3.1 Requirements collection

A requirement specifies a capability or condition that must (or should) be satisfied, a function that the system must perform, or a performance condition a system must achieve. Generally requirements are generated by the organization that is developing the system and often reflect multiple stakeholders. It is a common practise to group similar requirements for a system, element or component into a specification. Each specification contains multiple requirements, such as a systems specification that contains the requirements for the system, or the component specifications that contain the requirements for each component [41]. With reference to the V diagram described in [21], verification only confirms

compliance with the specification, while validation of the integrated system concerns its requirements, especially for type approval and certification purposes.

The output of the Requirement collection step is the identification of the system boundaries and of the key features which characterise the packaging machine.

3.3.2 Modular System Design

MBSE is a multidisciplinary approach that is intended to transform a set of stakeholder needs into a balanced system solution that meets those needs. It is a key practise to address complex and often technologically challenging problems.

The MBSE process includes activities to establish top-level goals that a system must support, specify system requirements, synthesize alternative system designs, evaluate the alternatives, allocate requirements to the components, integrate the components into the system, and verify that the system requirements are satisfied.

One of the powerful feature of the MBSE approach is the chance to synthesize alternative system designs.

Following the V diagram, a high-level system model is designed in the first step and the modules in the module-level inherit its information. For the highest level (the system level), the requirements are inherited from the "system context". This is composed by the system boundaries (all the actors and the quantities that interact with the system under development), the operational concept (the information exchanged with the external systems), and the stakeholder requirements (the needs of individuals or organizations that have direct interests) [48].

In accordance with the system requirements, the functions that the system must performed are determined. Eventually the system is subdivided into modules, one for each of the previously identified functions.

3.3.3 Implementation

This is the functional modeling, in which all the modular blocks are modelled in closed-loop control subsystems, each one consisting of Controller and Plant Model.

3.3.4 Testing

Testing is applied through HIL simulation, which provides, as it can be inferred in the V diagram, verification and validation (V&V) of the controller, performed in parallel to the machine development. In this phase, also called Virtual Commissioning, there is the release of virtual prototypes, with which HIL simulations are performed. If the V&V is

successful, i.e. it confirms compliance with the machine requirements, then the brand new HIL scheme proposed in this thesis is subsequently validated.

At this point, in the V diagram, System Performance Analysis & Simulations ends, and Prototype Development can start, with the Physical Integration step.

3.4 System architecture

In this section it will be described the implementation of the design model methodology presented in the previous section.

3.4.1 Requirements collection and Modular System Design in SysML

SysML is the language used for the first two steps of the development process, as reported in Fig. 3.5.

The SysML architecture is organized by hierarchy and consists of three level of abstraction: i.e. system, modules, and components levels. Following the V diagram, a high-level system model is designed in the first step and the modules in the modules-level inherit its information. Next, the modules are defined on a high level of abstraction and are broken down into components. Package diagrams are used to organize the system model and Fig. 3.6 is a view of an example of SysML model architecture applied to the system level. This pattern is applied to all elements of every level of abstraction. In each level hyperlinks were created to directly access to the selected view and to jump to the upper or lower level of abstraction [48].

An activity is created for every functional requirement, and all activities are ordered and connected, through control flow, in an activity diagram.

Eventually, the system is subdivided in modules, each module corresponding to one of the identified activity, which are represented as blocks with the allocated functions. The term functional allocation is a subset of behavioral allocation, and it refers specifically to the allocation of activities (also known as functions) or actions to blocks or parts [41]. Then it is associated a physical principle to the modules.

These processes are applied iteratively throughout the development of the system, with ongoing feedback from the different processes. Variants of the system design are applied recursively at each immediate level of the design, down to the level at which the components are procured or built.

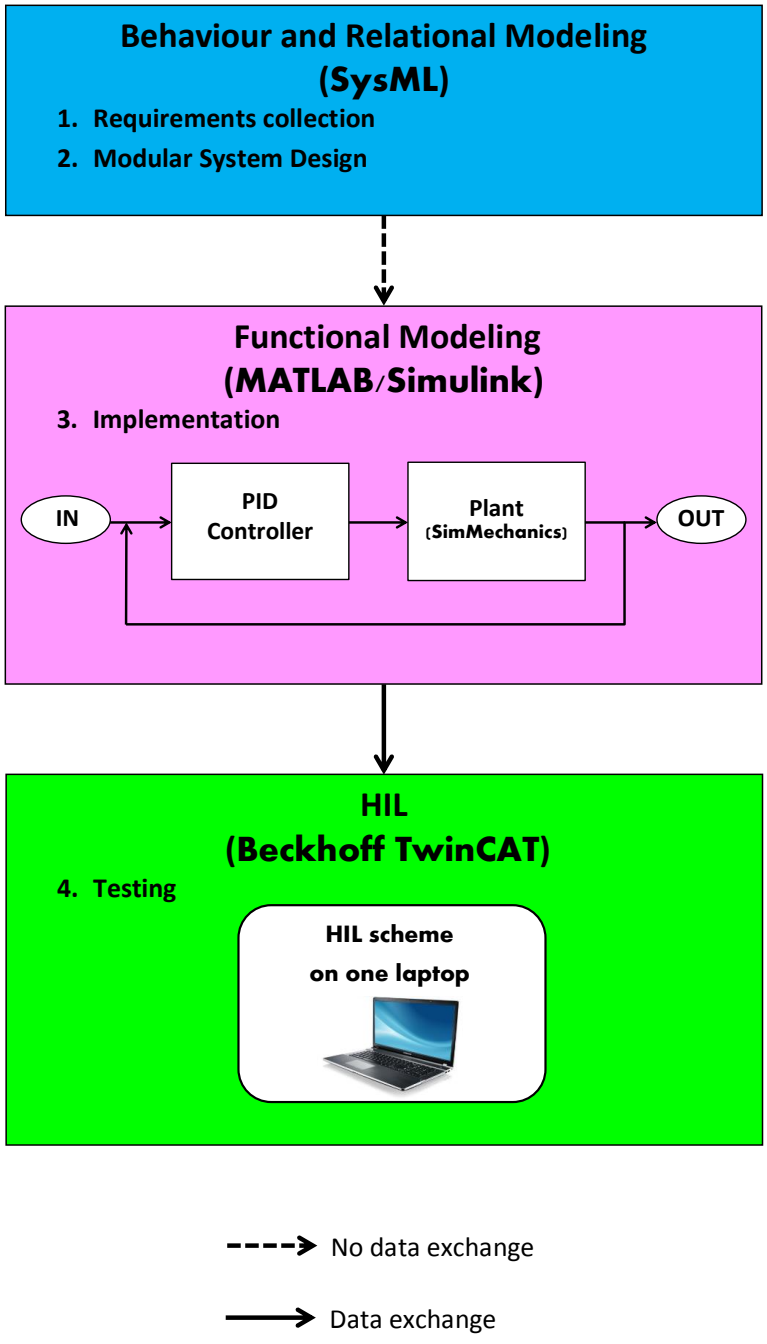

Figure 3.5: General scheme of the proposed Software Architecture

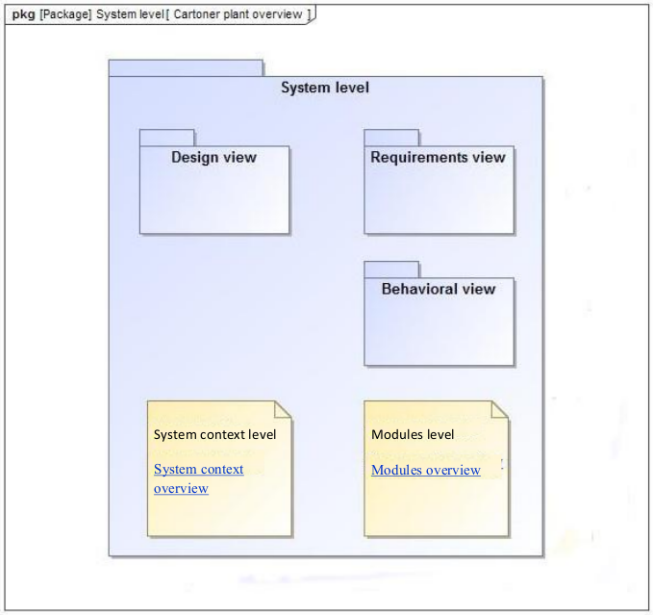

Figure 3.6: Organizing a Package Hierarchy

3.4.2 Implementation in MATLAB/Simulink

In a packaging machine the motion can be implemented using an electric servo motor controlled so as to ensure a desired motion profile which is necessary to achieve the desired functionality of the machine. In this case, the design phase of the kinematic or of the cam is translated in the preparation of a suitable program written for digital control of the movement of the electric motor. The motion profile should be generated in order to minimize the mechanical stresses of the component of the kinematic chain which can, in the long run, lead to damage of the overall system [38].

Servo systems act in closed loop systems, that includes:

- A computation unit (SW and HW, with control algorithms such as PID)
- An actuation (energy conversion and/or power modulation)
- A target system (that can be a mixture of mechanical, fluid, thermal, chemicals, electrical aspects)
- A feedback (energy conversion and/or signal processing)

In our workflow, each modular block identified in the previous subsection is modelled in MATLAB/Simulink following the control feedback scheme of Fig. 3.7. Each modular block needs its own motion profile, which is given to the input of the control feedback scheme. Firstly it is modelled the mechanical system via SimMechanics. If the output of

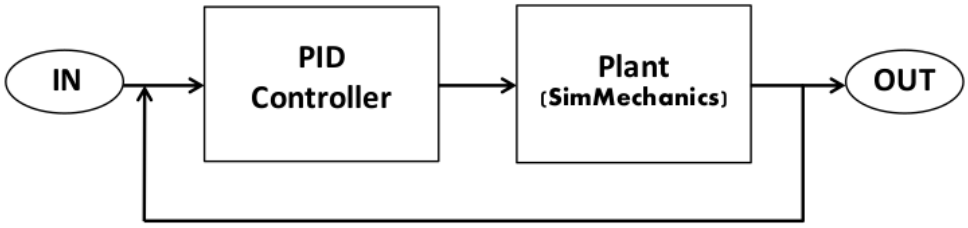

Figure 3.7: The control feedback scheme applied in MATLAB/Simulink for each modular block

the MATLAB/Simulink model does not track the motion profile, then it is necessary to insert a PID controller and properly tune it. When the reference results are obtained, it is necessary to shift to a Fixed-Step solver simulation for real-time viability.

The MATLAB/Simulink model is real-time capable if it meets both of these criteria when simulating it:

1. The results match expectations, based on empirical data or theoretical models.

2. The model simulates without incurring an overrun.

If Simulink simulation is successful, to this extent, for going from Matlab/Simulink to Backhoff TwinCAT platform, after having reconfigured the model for external inputs, the following steps have to be done in the Configuration window of the Simulink model:

1. Select TwinCAT Target

2. Simulink Coder generates C or C++ Code

3. Microsoft C/C++ Compiler generates Binary

4. TwinCAT Target generates description file tmc

This feature lying in the ability of automatically generating large amounts of code lines at one button touch and in a very small time interval, appears to be one of the main reasons for MBSE's growth in popularity. This step reduces radically time taken for code implementation in case of hand-written code used in traditional design methodology and avoids manually coded errors and bugs. Binary file can be used directly in TwinCAT. Next step will be to download the executable for the execution in the TwinCAT Runtime.

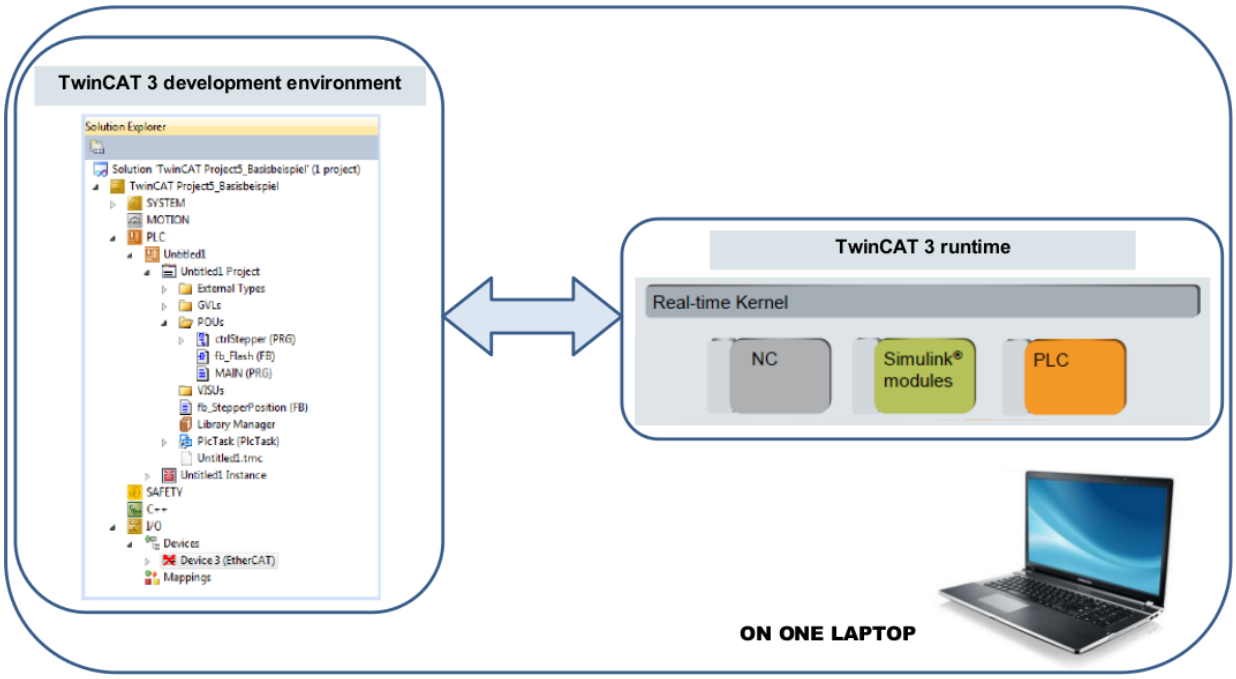

Figure 3.8: Architecture of HIL simulation with Beckhoff TwinCAT employing just one laptop

3.4.3 HIL testing

After having set proper parameters for the Simulink model in MATLAB/Simulink environment, and generation of the C++ Code, then HIL simulation can be performed, with a PLC executing the logic code of the modular block.

For configuring motion control, TwinCAT NC PTP allow to realize point-to-point movements in software.

Numerical control (NC) is the automation of machine tools that are operated by precisely programmed commands encoded on a storage medium, as opposed to controlled manually by hand wheels or levers, or mechanically automated by cams alone. The axes are represented by axis objects and provide a cyclic interface, e.g. for the PLC. This axis object is then linked to a corresponding physical axis. In this way the most diverse axis types with the most diverse fieldbus interfaces can be connected abstractly with the axis objects, which always offer an identical configuration interface. The control of the axes can be configured in various constellations (position or velocity interface) and various controllers in TwinCAT Engineering.

The virtual HIL environment integrates PLC, Motion Control, and Simulink/SimMechanics models on a unique CPU, and allows use of an ordinary personal computer for the HIL simulation of a module of a packaging machine, without any other hardware tool. This new HIL scheme is employed both for the real time execution of the plant modular

model and the control system, in just one laptop, as illustrated in Fig. 3.8 [4].

The parameterization in the TwinCAT System Manager is up-to-dated accordingly to the modular plant block under test in HIL scheme. *Ad hoc* test vectors are created as stimulus inputs.

The simulation is successful when the output signals reproduce the expected curve, following the same trend of the corresponding Simulink simulation.

3.5 Discussion

In this chapter it has been presented the procedure followed in the present research work for the MBSE design and verification of a packaging machine. The development process terminates with the testing via HIL modular simulations. Such techniques are becoming more commonly used, thanks to their specialized tools that simplify and enable the integration of the model of the physical system with that of the control system.

Modular blocks of complex mechatronic system can be encapsulated into separate Simulink models. By encapsulating the sub-systems, TwinCAT modules can be generated from each of it. These modules are used for the real-time simulation of the sub-systems in TwinCAT. The real system is put into operation successively, while individual system components are still integrated as simulation models.

With this simulation technique, we can easily implement controller designs on PLC hardware, and, since the design time is reduced, then it is now worth pursuing new business opportunities for low-volume products.

The objective of the HIL simulations here is even the verification of a brand new HIL scheme via the validation of the modular blocks of the packaging machine. The modularization approach is extended at HIL level, employing an HIL environment which runs real-time executions of Simulink/SimMechanics models into Beckhoff TwinCAT software.

The complexity of mechatronic systems has dramatically increased in recent years, due to the competitive demands and technological advances. Specifically, many products incorporate the latest processing and networking technology, which has significant software content with substantially increased functionality. These products are often highly interconnected with increasingly complex interfaces. Therefore it is become imperative to employ, in the machine design and production, hardware and software components which guarantee interoperability, scalability, sustainability, safety, and cyber security.

Concerning the softwares tools employed in the proposed System Architecture, I found in two of them the following drawbacks:

- With reference to MATLAB/Simulink: I needed to switch from Simscape to State-

flow for one the module that will be described in Chapter 4, since even if I followed advices from MathWorks for the solver configuring, nevertheless the simulation did not work moving to Beckhoff TwinCAT. Even if it was correct in MATLAB/Simulink.

- With reference to Beckhoff TwinCAT: since TwinCAT lays on Microsoft Windows, it has to chase its software updates. For instance, shifting from Windows 8 to Windows 8.1 has resulted in the need to repeat enabling the test-certificate mechanism for implementing TwinCAT modules. Furthermore, updating the TwinCAT versions sometimes has brought to several changes into the application and, as a consequence, time has been lost to restore the proper functioning of simulations, which were correctly running in the earlier versions.

Chapter 4

Case studies and experimental validation

4.1 Introduction

This chapter describes the implementation of the control strategies and the software architecture presented in the previous chapters to address two case studies.

The goal is to validate a brand new HIL scheme on modular mechatronic structures. This has been done implementing a new development platform, based on PLCs, and performing HIL simulations by means of just one laptop.

Firstly the Jaw System Unit (JSU) HIL project will be presented, with the aim to exploit the expansion of possibilities of TwinCAT development platform.

The HIL simulations have been preliminary accomplished on JSU to minimize the impact of the new platform, by adopting a fully tested solution. Indeed, we have reused a SimMechanics model of JSU, a packaging device which is currently employed by several Tetra Pak filling machines all over the world.

Then, the chapter presents the case study of a cartoning machine design project, which has been addressed more intensively, in order to emphasize the potential benefits of the proposed technology.

In this second case study, the whole design and virtual testing workflow, which has been illustrated in the previous chapter, will be implemented following the MBSE paradigm.

Experimental tests performed for validating either the JSU and the cartoning machine design will provide a proof of concept of reusability for the HIL scheme.

At the same time, the application of the new HIL scheme to the mechatronic structures of the cartoning machine, will implement modularity and reconfigurability of the proposed software architecture.

4.1.1 Outline

The outline of the chapter is as follows: Section 4.2 describes the Jaw System Unit HIL project and its results validating the brand new HIL environment presented in this thesis. Then, Section 4.3 describes a packaging machine design project that provides a proof of concept of reusability and reconfigurability of the HIL scheme at modular level. This second case study implements the whole software architecture proposed in the previous chapter.

4.2 Jaw System Unit (JSU)

4.2.1 Software architecture

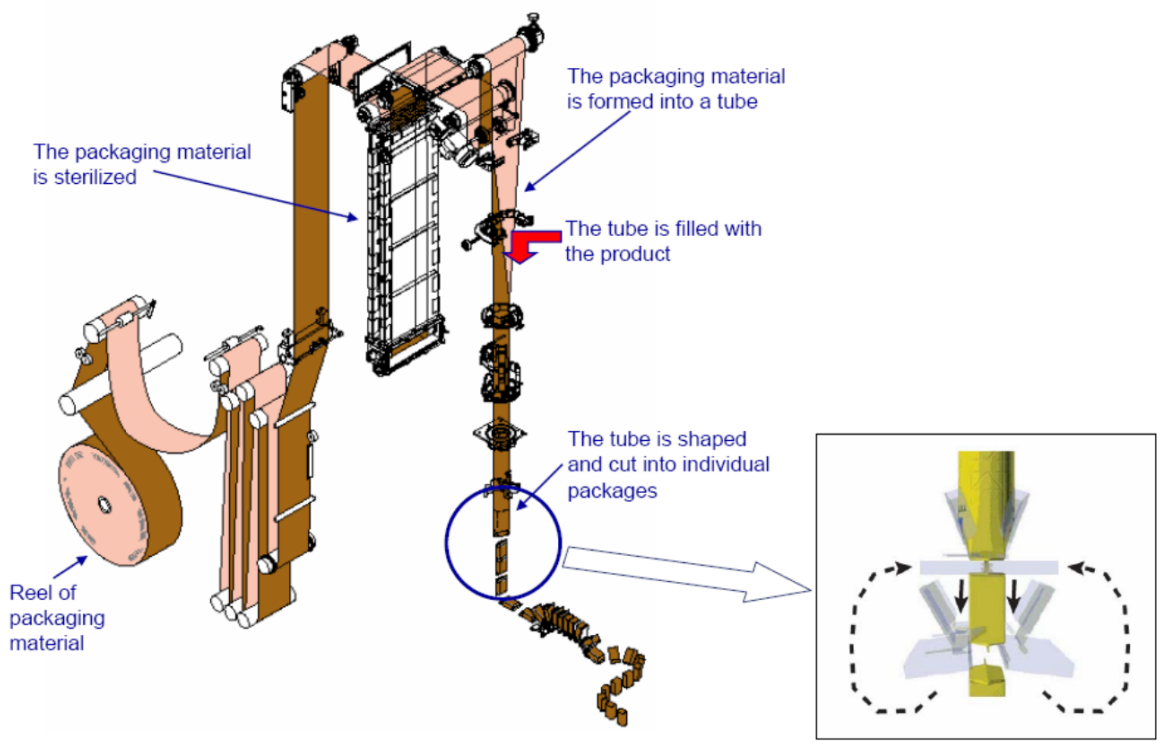

Figure 4.1: Tetra Pak filling machine. In the square frame Yoke and Jaw motions are outlined. (source: Tetra Pak)

In Fig. 4.1 is shown a Tetra Pak filling machine, a mechatronic system belonging to the primary packaging level of a packaging line.

Tetra Pak is a multinational company which provides processing, packaging, and distribution solutions for food products [39].

Within filling machine, web is sterilized and then formed in a cylindrical continuous

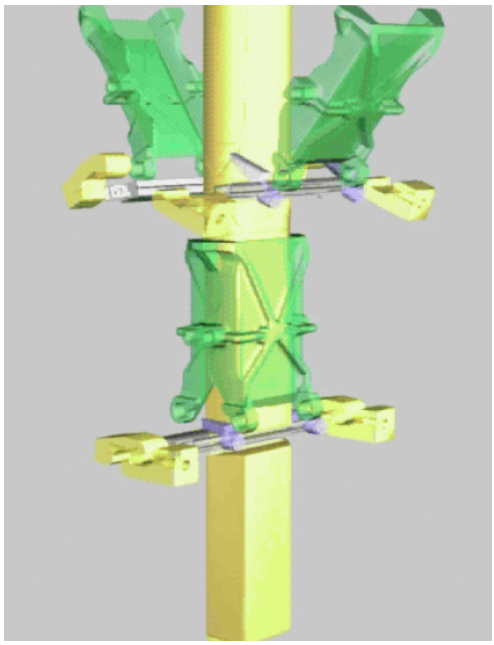

Figure 4.2: Package forming (source: Tetra Pak)

shape, called web tube. The product is inserted in the tube by the filling system and then the tube is shaped and cut into individual packages.

In this filling machine the Jaw System Unit (JSU) is the device which performs the package forming sequence via a couple of two jaws, which seal and cut the web, providing the package shape.

JSU has two symmetric sides, a left and a right one, and each side has two jaws.

Furthermore each side of JSU has two servo motors, corresponding to the two degree of freedom Yoke and Jaw, which are illustrated in the square frame in Fig. 4.1. By means of Yoke and Jaw axes, each couple of jaws perform an alternating up and down motion, giving together rise to the package forming process, as illustrated in Fig. 4.2.

In Fig. 4.3 is shown just one side of JSU and its main components.

Being the two sides of JSU specular, in this case study we will focus on just the Left JSU side.

The design methodology applied in this case study concerns just the Implementation and the Testing blocks in the Software Architecture outlined in Fig. 3.5. Our objective, in fact, was to implement and validate the Beckhoff TwinCAT platform, rather than re-engineering the package forming device.

By employing the JSU, the aim was to minimize the impact of the new HIL platform.

Actually the JSU is a fully tested packaging device, the result of years of optimizations. Currently it is still deployed by several Tetra Pak filling machines all over the world.

Therefore firstly a Simulink/SimMechanics model has been designed, reusing an al-

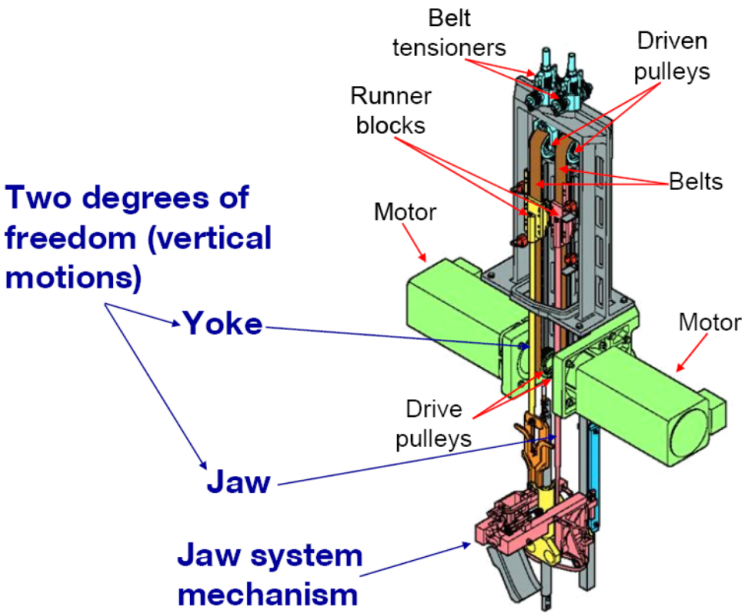

Figure 4.3: A JSU side and its main components (source: Tetra Pak)

ready available SimMechanics model of JSU.

Then, after the C++ code generating, it has been performed an HIL simulation in Beckhoff TwinCAT environment.

Simulink/SimMechanics model

As reported in Fig. 3.7, the Simulink model consists of SimMechanics model for the machinery and PID blocks for the relative controller.

A SimMechanics block was obviously already available, since JSU is a fully tested device, we just needed minor changes to convert the mechanical model from SimMechanics First to SimMechanics Second Generation, the latest version.

Control action is given by just P controllers, i.e. K_P in the Position Loop and K_V in the Velocity Loop, both in the Yoke and in the Jaw input.

The PID blocks have been automatically tuned via the Control System Toolbox, a MATLAB product for modeling, analyzing, and designing control systems.

The Simulink model of the LEFT JSU is shown in Fig. 4.4.

It has two inputs and four outputs:

- YOKE_CmdPos: yoke position input
- JAW_CmdPos: jaw position input
- YOKE_PositionFB: yoke position output

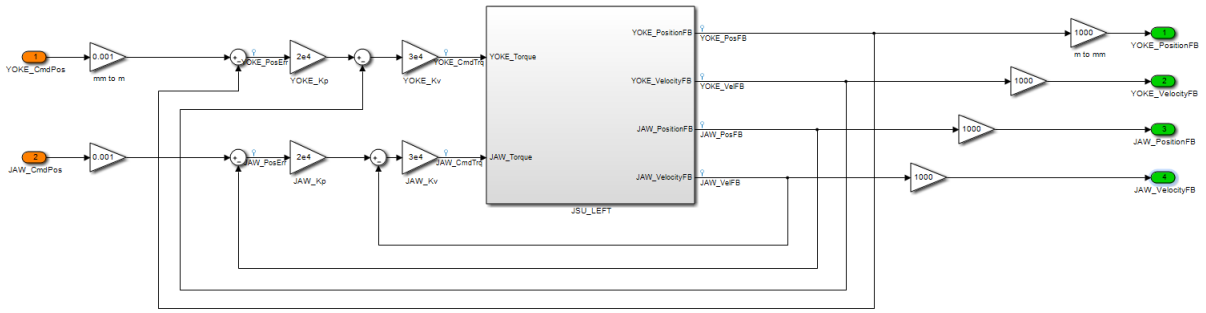

Figure 4.4: Simulink model for JSU

- YOKE_VelocityFB: yoke velocity output
- JAW_PositionFB: jaw position output
- JAW_VelocityFB: jaw velocity output

Inputs signal Inputs values for YOKE_CmdPos and JAW_CmdPos signals have been supplied via a MAT-file (i.e. a binary MATLAB file that stores workspace variables), and FromWorkspace blocks in the Simulink model.

The MAT-file has been provided by Tetra Pak.

Configuring Simulink parameters

Solver You simulate a dynamic system by computing its states at successive time steps over a specified time span. This computation uses information provided by a model of the system. The time steps are time intervals when the computation happens. The size of this time interval is called the step size. The process of computing the states of a model in this manner is known as solving the model. No single method of solving a model applies to all systems. Simulink provides a set of programs called solvers. Each solver embodies a particular approach to solving a model.

A solver applies a numerical method to solve the set of ordinary differential equations that represent the model. Through this computation, it determines the time of the next simulation step. In the process of solving this initial value problem, the solver also satisfies the accuracy requirements that you specify.

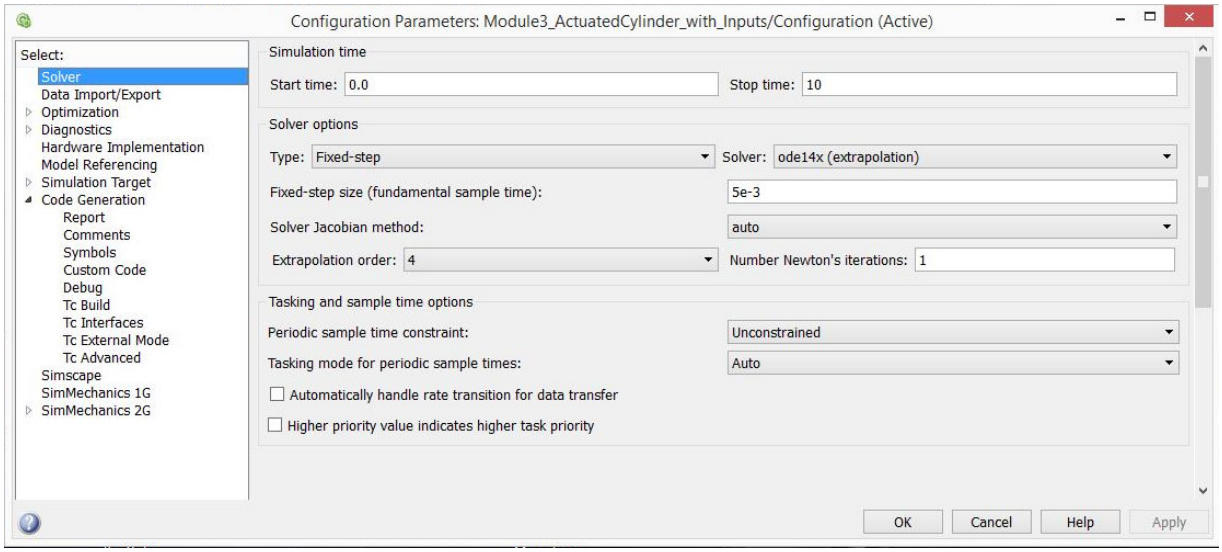

Figure 4.5: Solver pane for a Simulink model's active configuration set

To run a model on a real-time target, it is mandatory to configure the model for fixed-step, fixed-cost simulation.

The step size and the number of iterations that you specify affect the computational cost of your real-time simulation. As you decrease the step size or increase the number of iterations, the results become more accurate, but the simulation costs more, so it can take longer to simulate. Simulation overrun occurs if the step size is too small, or if there are too many iterations for the solver to calculate a solution in a single real-time computational frame.

In fixed-step solver Step size remains constant throughout the simulation.

Fig. 4.5 is the snapshot of a Solver pane, which sets up a solver for a model's active configuration set. The Solver configuration pane allows you to specify other start and stop times for the currently selected simulation configuration. By default, simulations start at 0.0 s and end at 10.0 s.

It is noteworthy that Simulink simulation time and actual clock time are not the same. For example, if running a simulation for 10 s, usually does not take 10 s as measured on a clock. The amount of time it actually takes to run a simulation depends on many factors including the complexity of the model, the step sizes, and the computer speed [29].

For the Simulink modules of the Left JSU, the following configuration has been set:

- Simulation time: Start time = 0.0; Stop time = 0.9 s
- Solver options: type = Fixed-step; Solver = ode14x (extrapolation)
- Fixed-step size (fundamental sample time): 5e-3 s

- Solver Jacobian method=auto, Extrapolation order=4, Number Newton's iterations=1: default

Fixed-step solver ode14x (extrapolation) uses a combination of Newton's method and extrapolation from the current value to compute the model's state at the next time step, as an implicit function of the state and the state derivative at the next time step. In the following example, X is the state, DX is the state derivative, and h is the step size:

$$X(n+1) - X(n) - h * DX(n+1) = 0.$$

This solver requires more computation per step than an explicit solver, but is more accurate for a given step size.

Code generation Going from Matlab/Simulink to TwinCAT Module requires also a specific configuration to be set in the Code generation pane in Simulink, in order to export TwinCAT modules directly after the code has been generated.

In all the TwinCAT projects of this work research, Simulink has been configured with the following parameters:

- System target file = TwinCAT.tlc
- in Tc Build tab: Publish binaries for platform "TwinCAT RT (x64) is activated
- in Tc Advanced tab: task assignment = ManualConfig (to assign task(s) manually)
- in Tc Advanced tab: CallBy = Module (the module can be called from a PLC)

At this point it is possibly to click "Build Model" button to start code generation.

The generation process delivers three outputs [26]:

- C++ code generated by the Simulink Coder in the form of a VS C++ project. The generated project contains all source code files, compiler and linker settings that are necessary to successfully compile the module within TwinCAT environment.
- Binary (object file) produced by the Microsoft VS C++ compiler. Binaries can be added as TcCOM objects to the VS solution under the TwinCAT System. The inputs and outputs of the TwinCAT module are matching the ones in the Simulink model from where it was generated.
- description file .tmc generated by TwinCAT Target.

If everything is correct, now we can open the TwinCAT development environment and create a new TwinCAT project.

Beckhoff TwinCAT project

TwinCAT software provides deterministic cyclic access to inputs and outputs as well as to variables in the traditional IEC 61131-3 languages, therefore it suites at best the key requirements for HIL projects.

For employing TwinCAT in our research activities, there are a sequence of steps to be accomplished, in order to properly exploit TwinCAT functionalities.

MATLAB/Simulink enables the code generation from Simulink models through the Simulink Coder. With Simulink Coder plus the TwinCAT Target for MatLab/Simulink (TE1400), supplementary software from Beckhoff, it is possible to generate C++ code encapsulated in a standard TwinCAT module format that, in turn, can be loaded into the TwinCAT development environment.

Therefore after having customized coder settings, and triggered the building process within Simulink, for using the generated TwinCAT module, in the TwinCAT development environment you have to expand the node SYSTEM in the Solution Explorer, and then mouse click the menu item *Add new item* in the context menu of node TcCOM Objects.

The generated module can then be selected from the group TE1400 Module Vendor → Generated Modules.

Node SYSTEM is needed for the integration of the Simulink generated model in TwinCAT, and is handled for all our HIL projects.

PLC node as well is employed in all our TwinCAT projects, since we deployed PLC for running HIL simulations.

In case, even MOTION node is handled.

Fig. 4.6 shows an example of a Simulink model integrated in TwinCAT real-time environment, where it can be seen that the TwinCAT block diagram keeps the same layout and Simulink model element names.

In Fig. 4.6 can also be seen that, in TwinCAT XAE environment, MOTION and PLC configuration nodes are visible in the tree view of the System Manager.

One of the most interesting capabilities of TwinCAT is the possibility to display and navigate through the Simulink block diagram including the display of parameters and signals values that can be monitored and/or modified at run-time. The user can adjust these parameters within the TwinCAT environment without to change the original model.

In MATLAB/Simulink environment, before generating the code, Simulink input blocks for models, like FromWorkspace, Signal Builder, Repeating Sequence or Step blocks, are removed, because in TwinCAT we have to employ specific inputs.

In TwinCAT models signals inputs are provided via .csv (comma-separated values) files which have two columns, one for the time and the other one for the relative values.

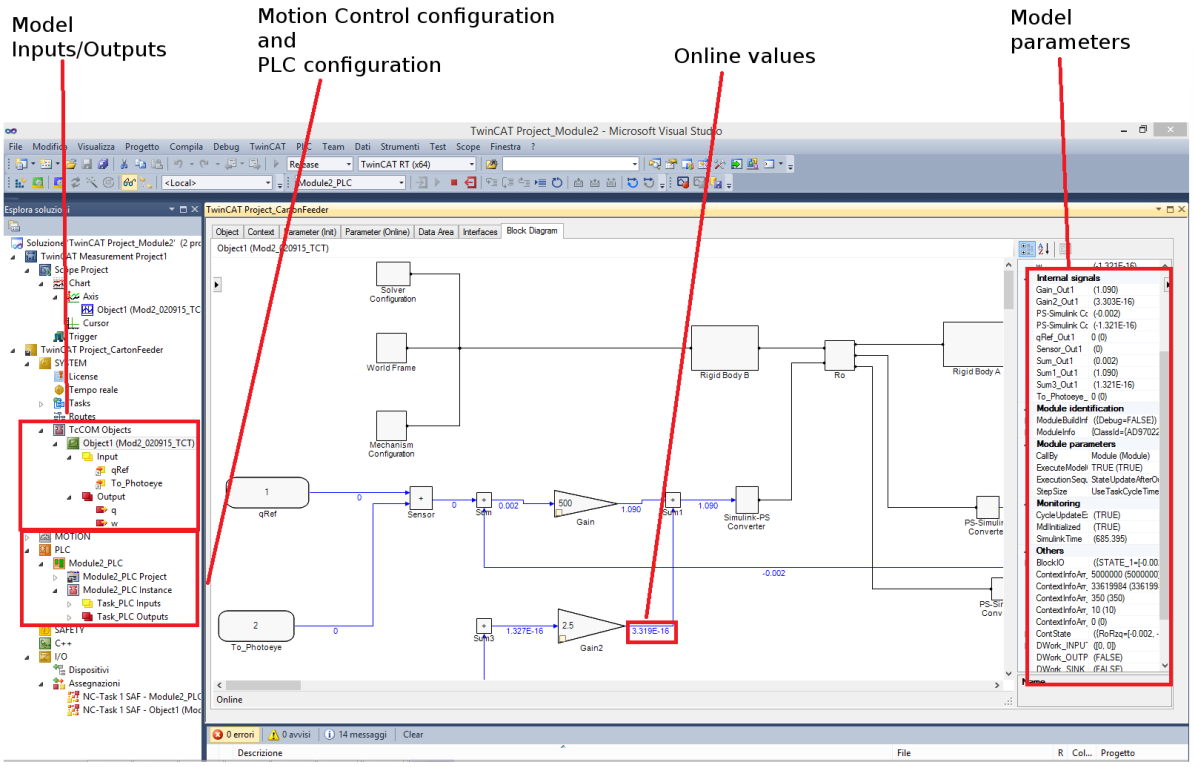

Figure 4.6: A Simulink model integrated in TwinCAT real-time environment

In the following, the steps to implement the HIL simulation for Left JSU will be described.

MOTION In MOTION you can create the required axes and set their parameters. It is available a CAM Design Tool for the Configuration of cam plates, i.e. for designing the movements of cam plates. The CAM Design Tool can be found in the System Manager at MOTION, under the Tables item.

It is possible here to insert additional masters, and to enter corresponding slaves under them. The procedure for developing a design of a cam involves the insertion of the .csv (comma-separated values) input files of the model in the relative table.

Left JSU has two input .csv files, one for YOKE_CmdPos (see Fig. 4.7) and the other one for JAW_CmdPos (see Fig. 4.8), with the same values as in in the MAT-file.

PLC PLC projects can be added in the PLC node of the tree view of the System Manager.

In Left JSU, the PLC module consists of two Program Organization Units (POUs), the MAIN program and the functional block FB_JSU_a060314_L. The FB_JSU_a060314_L

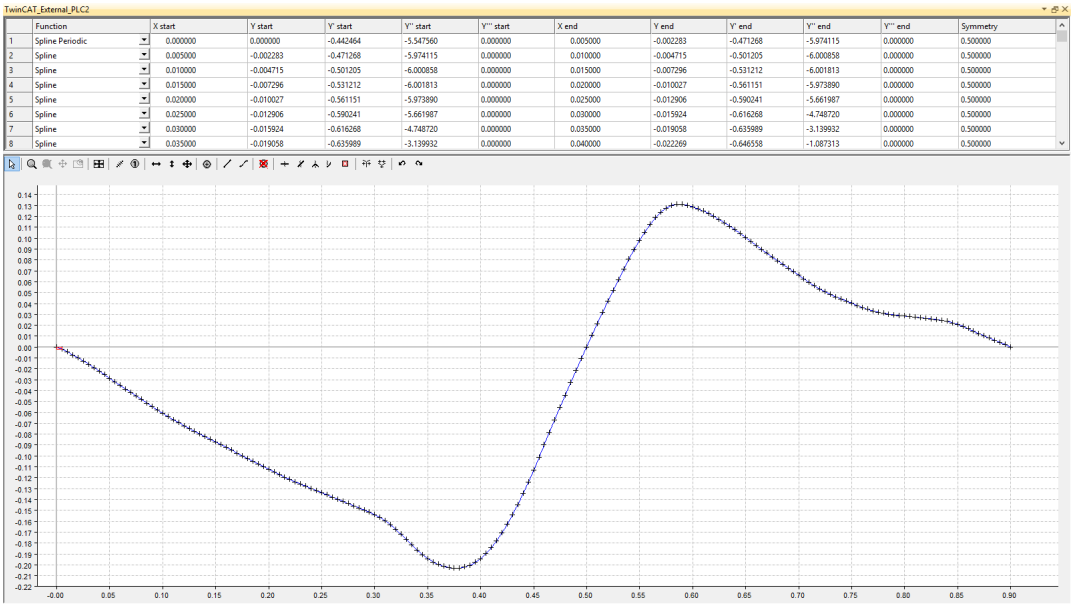

Figure 4.7: MOTION - Visualization of input YOKE_CmdPos in TwinCAT CAM Design Tool

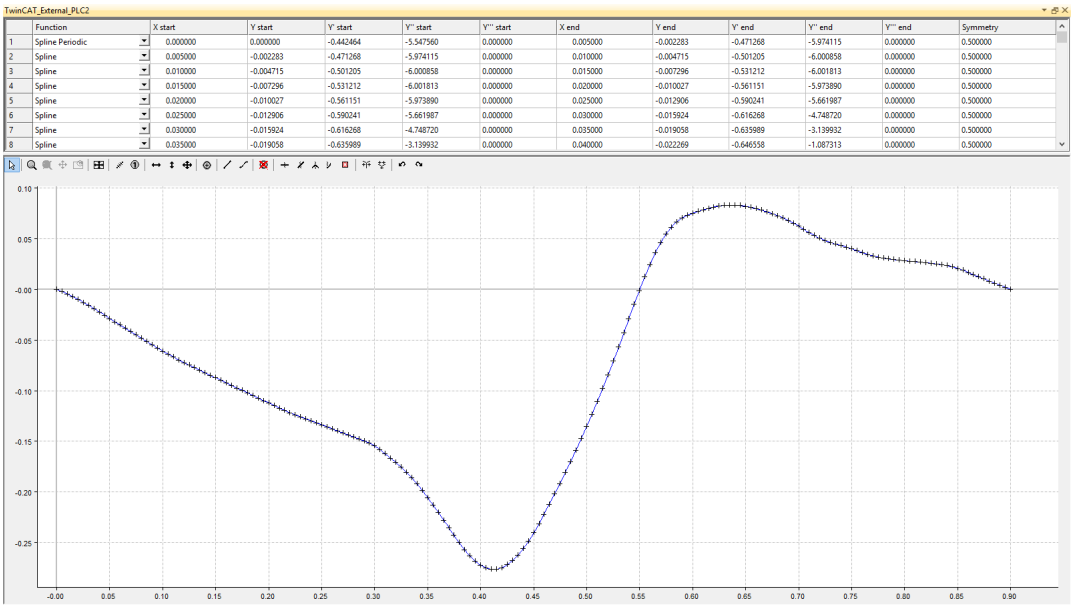

Figure 4.8: MOTION - Visualization of input JAW_CmdPos in TwinCAT CAM Design Tool

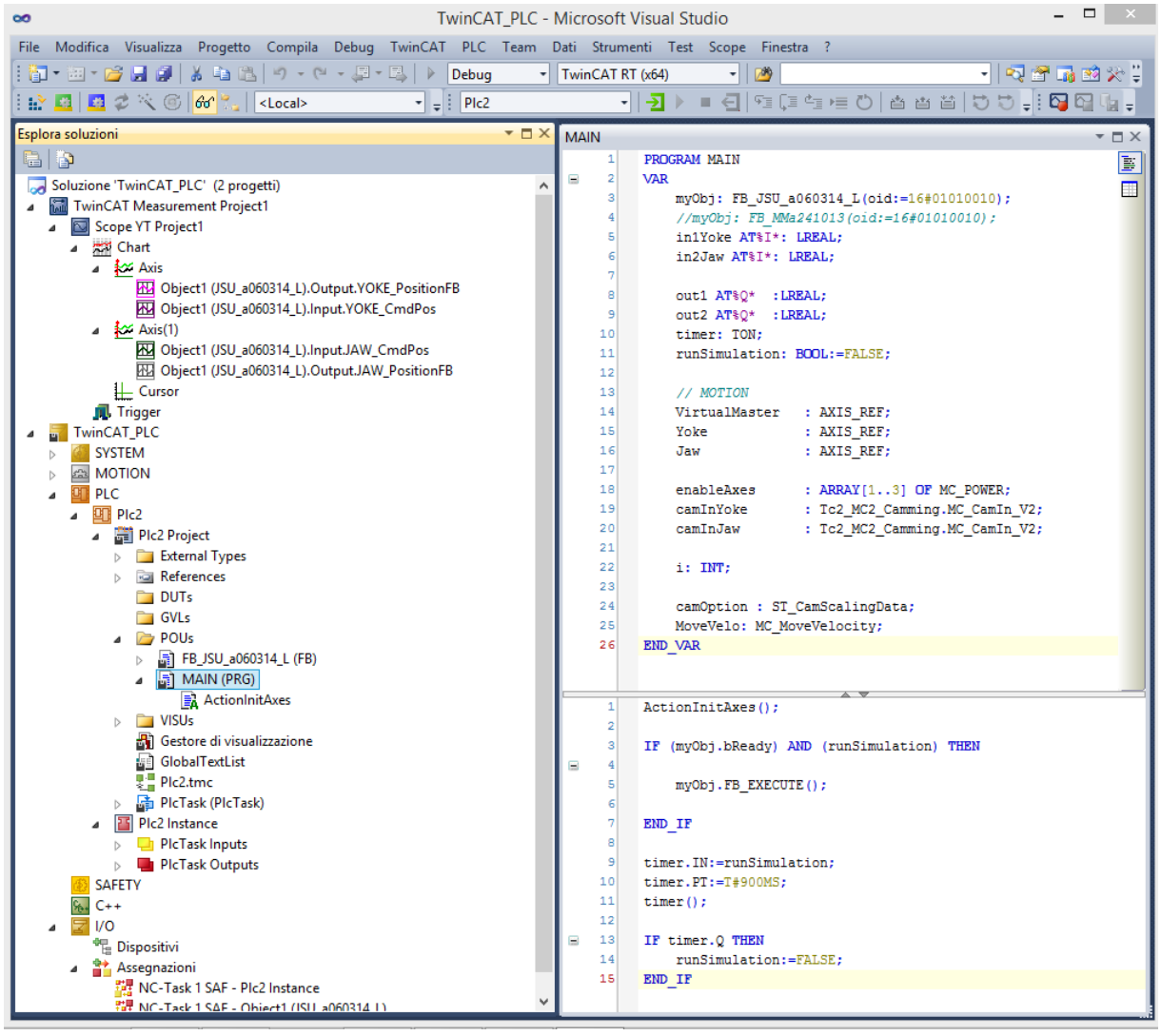

Figure 4.9: JSU PLC implementation

implements a set of methods and the state machine handling (see Fig. 4.9). An Action, i.e. an additional POU implementation which is assigned to the MAIN program, handles the motion control. The MAIN program instantiates the FB_JSU_a060314_L and a variable *runSimulation*, in order to repeat cyclically the simulation until the Stop of PLC is clicked.

4.2.2 Experimental validation for the Left JSU

Fig. 4.10 and Fig. 4.11 are charts of, respectively, the Left JSU Yoke position output and Jaw position output in Simulink simulation.

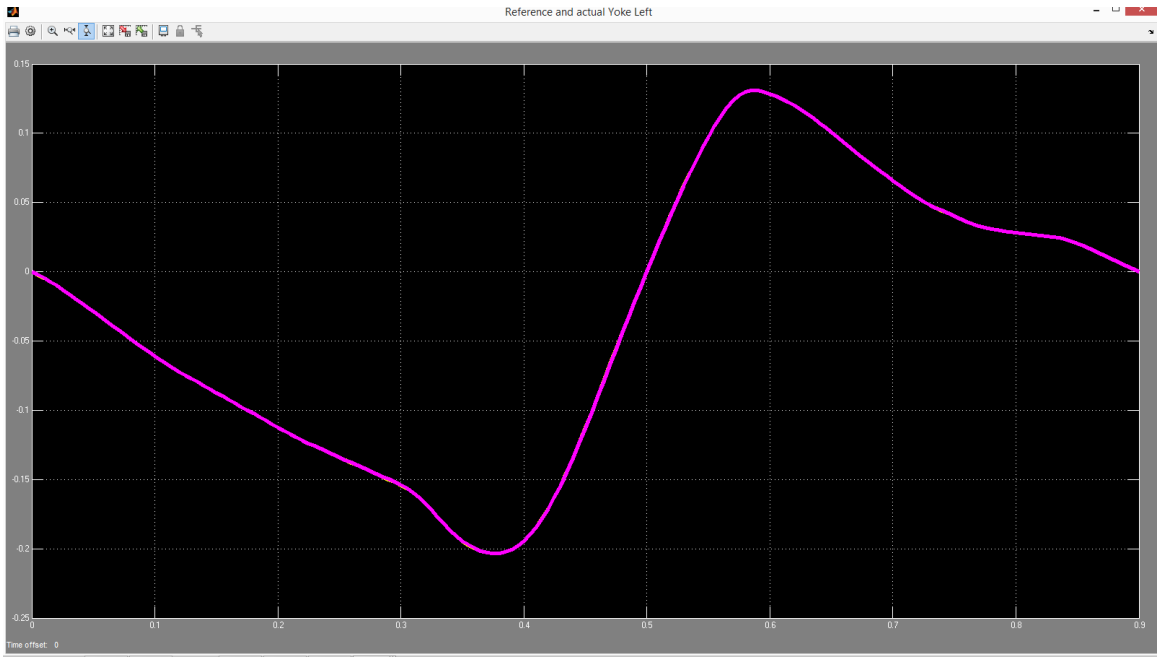

Figure 4.10: Reference and actual Yoke output signal in the Simulink simulation

Both the two images show the trend of the reference signal and the relative output signal. Since the outputs perfectly track the reference values, it can be derived that the Simulink simulation was successful.

Fig. 4.12 is a snapshot of the scope for the Left JSU HIL simulation. Yoke and Jaw outputs are represented stacked, in particular Yoke above in pink, and Jaw below in green.

Code in the Left JSU PLC MAIN program provides to repeat cyclically the HIL simulation, until PLC Stop is clicked.

Also in Fig. 4.12 charts report reference signal and the relative output signal both for Yoke position and Jaw position. Reference and output signals overlap perfectly either in Yoke graph and in Jaw graph.

In addition the trends of both Yoke and Jaw HIL outputs are coherent with the trends of, respectively, Yoke and Jaw Simulink simulation outputs.

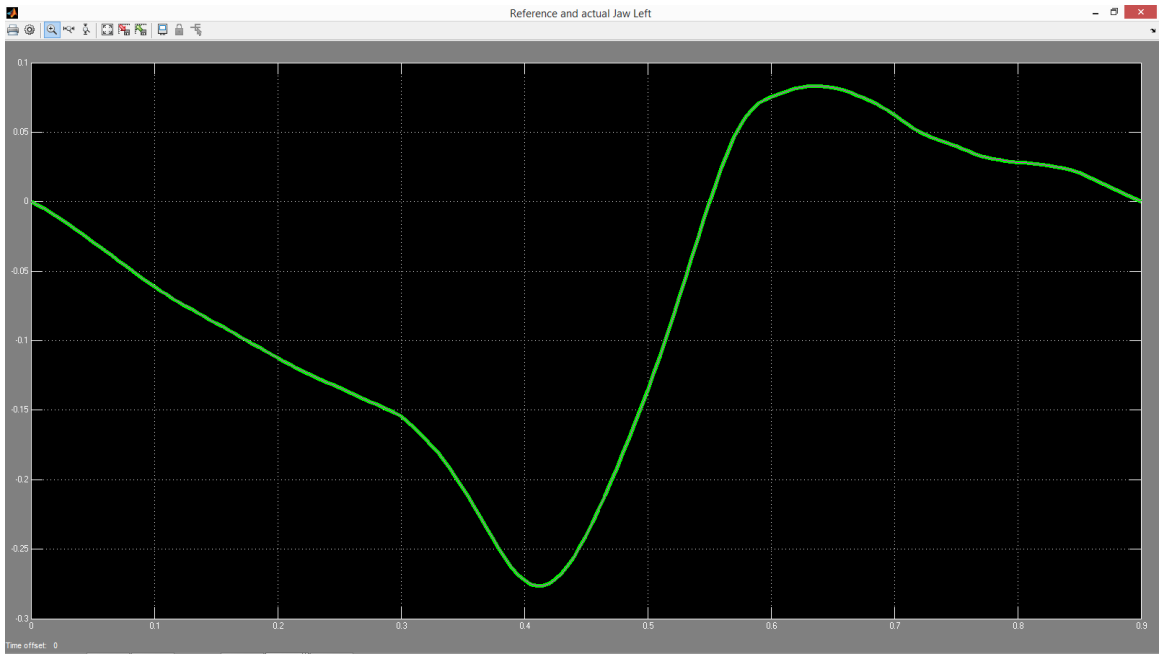

Figure 4.11: Reference and actual Jaw output signal in the in Left JSU Simulink simulation

Therefore the results of the Simulink simulation and of the HIL simulation are satisfactory, because both of them reproduce correctly the position of the Left JSU Yoke and Jaw outputs.

The HIL simulation is even a virtual commissioning of the Left JSU, since the TwinCAT simulation has been performed via a real PLC.

Moreover the brand new HIL scheme built on one just laptop is validated, since results of the HIL simulation applied to the Left JSU are correct.

The Left JSU HIL has been also successfully applied even with simulation in a 32 bit operating system Virtual Machine, and visualization in 64 bit operating system.

Finally an External mode version of Left JSU HIL project has been successfully deployed. Using this mode, Simulink works only as a graphical front end without any background calculation, just like a kind of visualization. If the model was translated to a TwinCAT module using the appropriate settings, Simulink can connect to the TwinCAT module which is currently running in a TwinCAT real time context. Parameters which are modified in Simulink, are downloaded to the module online.

These two further Left JSU HIL versions, both on just one laptop as well, confirm the expansion of possibilities of the TwinCAT development platform for implementing simulation capabilities.

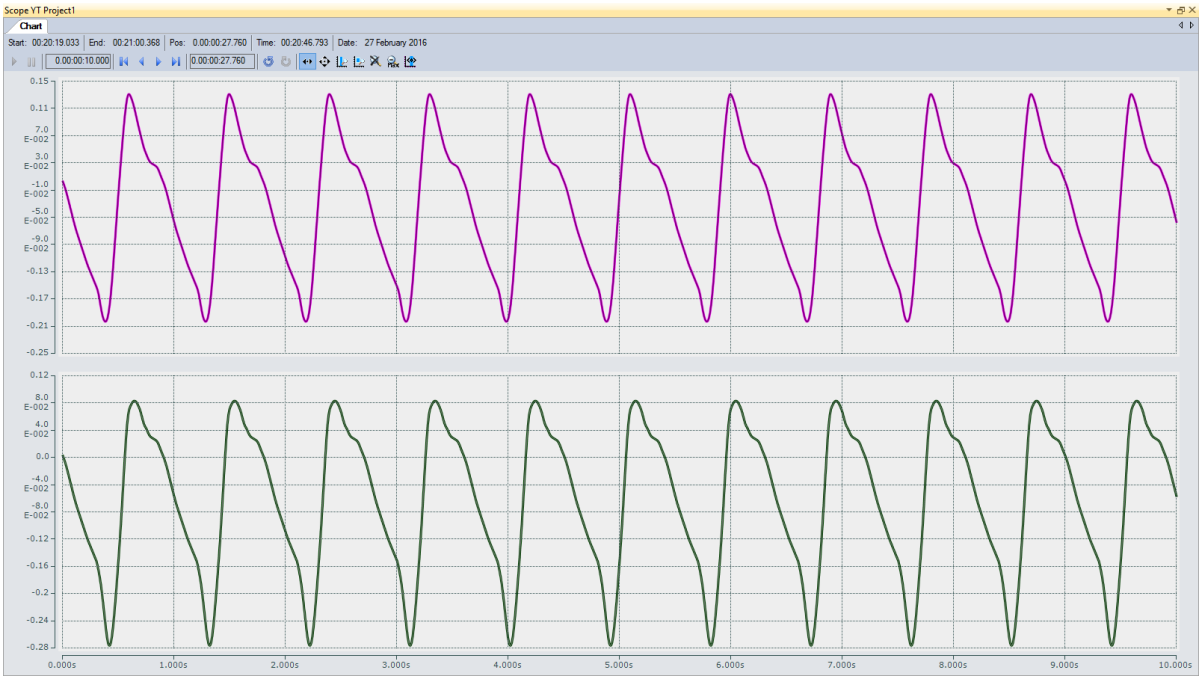

Figure 4.12: Reference and actual Yoke and Jaw output signals in Left JSU HIL simulation

4.3 Cartoning machine

With reference to the development process described in the previous chapter, a Cartoning machine (also called Cartoner) has been designed.

This is the case study used to validate the whole workflow and the new HIL scheme at modular level presented in this thesis.

Cartoning machines are a kind of machine belonging to the Secondary Packaging level in the production plants of countless industries. Cartons are available in a variety of styles, shapes and sizes. The package designer should try to standardize to one style whenever possible, in order to minimize the downtime for changeover of the automatic cartoning machine.

The Cartoner here described has been designed for the carton in Fig. 4.13 which is 19 x 9.2 x 6.4 cm and glue closed.

The basic carton is made at the converter by folding a die-cut piece of board into a four-sided tube glued together at the manufacturer's joint.

Flaps at the end of the tube are folded together for closure. Flaps may be either tucked or glued.

Glue carton have four flaps, two minors and two major flaps. After folding the minors, the inner major flap is folded, glue is applied and the outer major flap folded down top.

Cartoners are generally described by carton orientation and carton motion. Horizon-

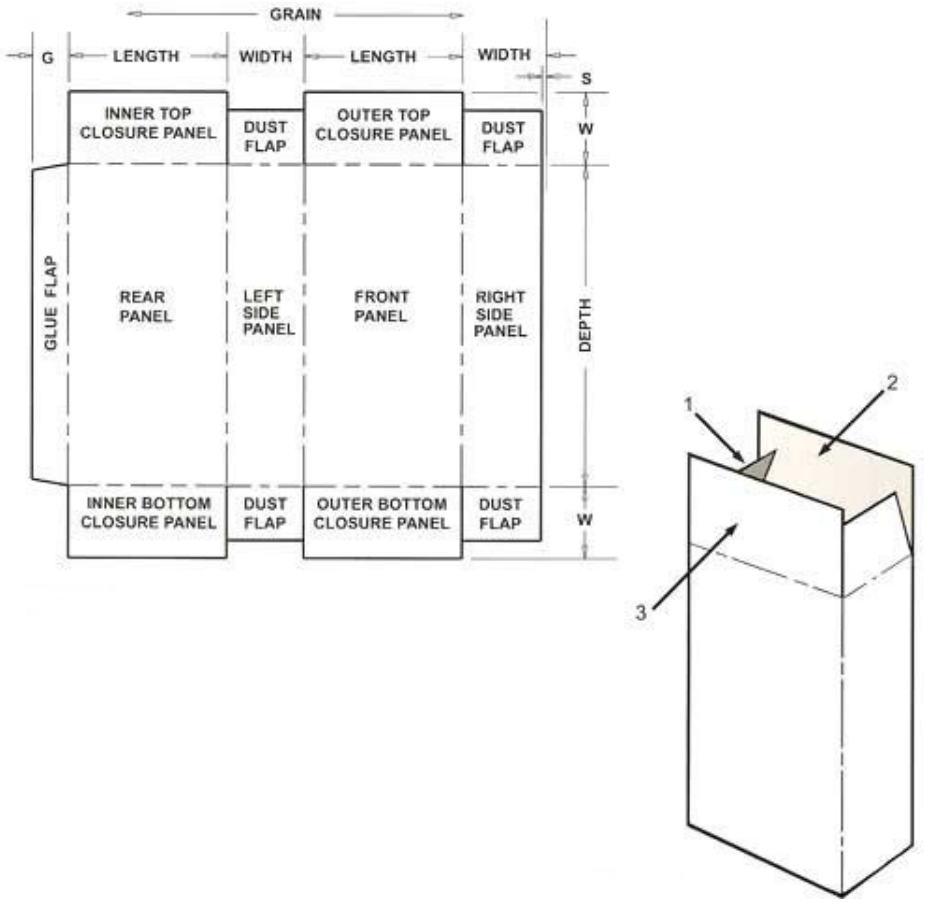

Figure 4.13: Straight glue carton (source: Institute of Packaging Professionals)

tal cartoners position the carton with the open ends to the side, perpendicular to flow. Horizontal orientations generally lend themselves better to automatic loading and higher speeds, while vertical cartoners tend to be better for lower speed manual loading. There are enough exceptions to this statement that it must be viewed as a generalization, not a hard and fast rule.

Cartoners are also described as intermittent or continuous motion. Intermittent-motion machines stop the carton momentarily, which can be of great help when manually loading multiple or more difficult products. They are generally lower speed, less than 60 to 90 packages per minute, and often vertical. Continuous-motion cartoners run at higher speeds ad are often horizontal.

Typically cartoning machines pick up a blank knocked-down carton from a carton magazine and erect it, then horizontally or vertically fill the carton through the open end with product, and finally close the open flaps [37].

The design methodology applied to the cartoning machine in this work gradually goes from high-level requirements with SysML, through system design with Simulink/SimMe-

chanics, and, after generating C++ code from the relative Simulink model, into low level platform specific embedded prototype simulations, with Beckhoff TwinCAT.

4.3.1 Software architecture

Development of the SysML Project

The SysML modeling architecture follows the design pattern in the SysML block of the software workflow proposed in the previous chapter.

The designing of the cartoning machine has been developed from scratch.

To identify the set of desired system requirements for the packaging machine, it has been analysed a video of an operating cartoner recorded in a production plant [49]. The resulting Requirements table is reported in Fig. 4.14.

#	Id	Name	Text
1	1	Original statement	Design a system for inserting products into cartons.
2	1.1	Infeed conveyor	An Infeed conveyor provides for transferring products suitably spaced on a bucket chain.
3	1.2	Carton feeder	It shall pull a blank carton out of the carton magazine and erect it.
4	1.2.1	Pickoff	Pulling a blank carton out of the carton magazine, if there is the command from the corresponding proximity sensor.
5	1.2.2	Positive carton opening	Opening the blank carton.
6	1.3	Loading unit	It shall fill the carton through the open end with product.
7	1.4	Carton closing	It shall finally close the open flaps through either an adhesive or tuck method.

Figure 4.14: Requirements table of the Cartoner SysML Project

Then the following specifications have been chosen:

1. Horizontal cartoner
2. Continuous motion
3. The product pockets are conveyed along to the point from where the products are loaded into a single carton. Therefore there are two parallel conveyors, i.e. the infeed conveyor, which tranports the products, and the carton conveyor, where cartons are placed after their opening.
4. Speed: 160 cartons per minute (= crt/min)

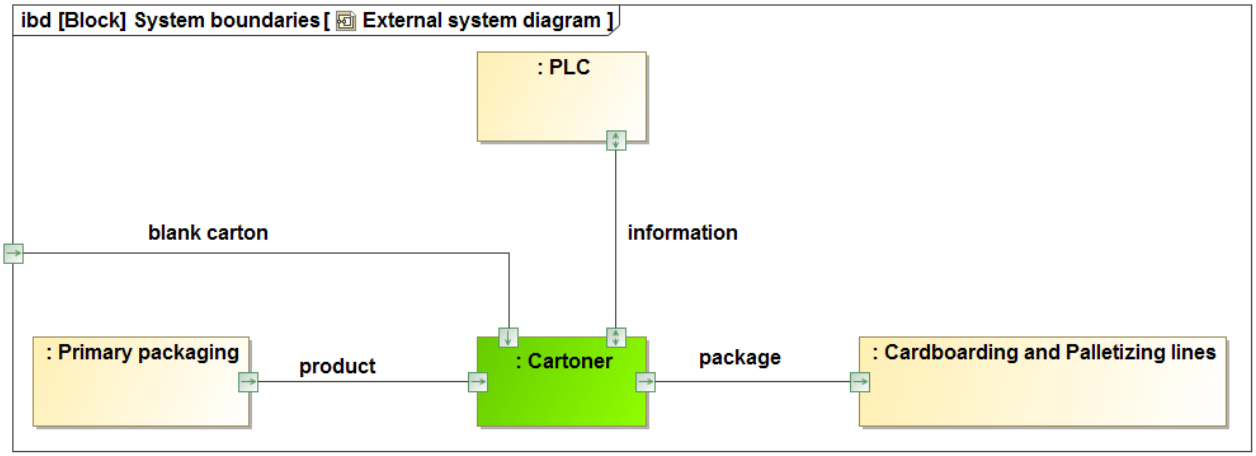

Figure 4.15: Representation of the systems (in apricot) which interact with the designed machine (in green)

These specifications are consistent with the characteristics of the machine shown in the video. The first two are typical of fully automated machines.

The speed was instead found out referring to technical data of a similar automated cartoner already in the market [50].

Starting from the collection of the system requirements, it is mandatory to identify what is external to the system that may either directly or indirectly interact with it. The Internal Block Diagram (ibd) in Fig. 4.15 defines the Cartoning machine boundary and the external systems. It is also represented the relation between PLC and the cartoner, while interactions of PLC with other machines in the External system diagram are not shown.

From the requirements analysis, the system functional architecture was individuated and the system was decomposed into four modules, assigning them the functions they must fulfil:

- Module 1: "Provides for transferring products suitably spaced on a bucket chain"
- Module 2: "It shall pull a blank carton out of the carton magazine and erect it"
- Module 3: "It shall fill the carton through the open side with product"
- Module 4: "It shall close the open flaps through either an adhesive or tuck method"

An activity is then created for every functional requirement, and all the activities are ordered and connected, through control flow, in an activity diagram. See Fig. 4.16.

In this phase the system requirements have been broken down to the modules requirements and it has been associated a physical principle to the modules:

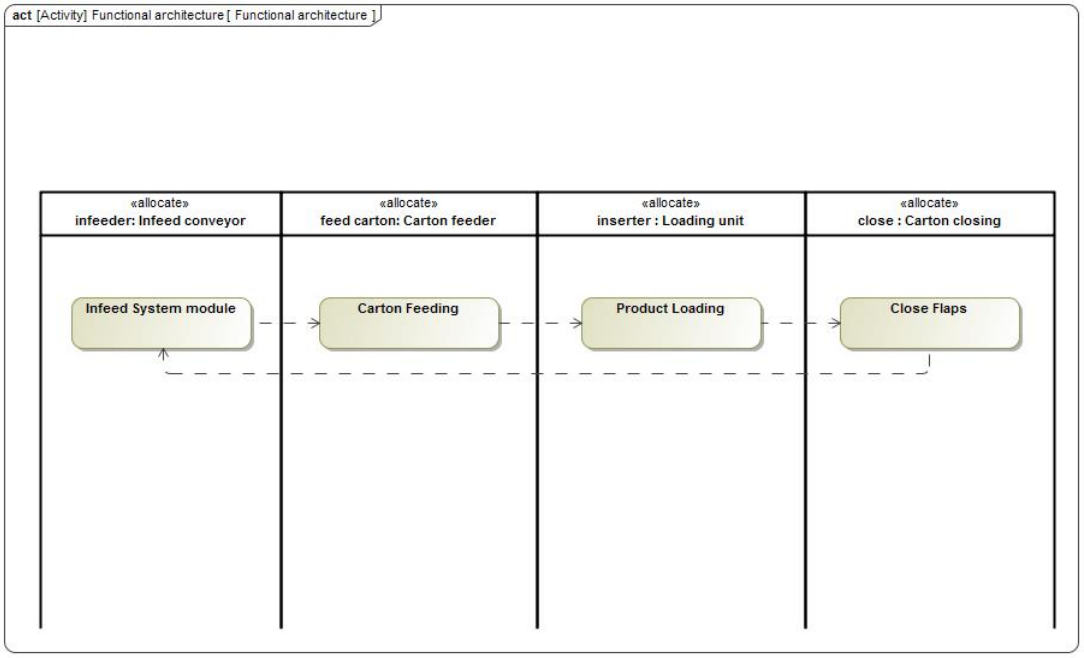

Figure 4.16: Activity Diagram

- Module 1: "a product lug pocket conveyor"
- Module 2: "an arm of 1 Degree of Freedom (DoF), provided with two suction pumps, picks off a blank carton at 45°, rotates, and puts it on the loading conveyor. The opening of the carton is done by metal blades that outcome from the loading conveyor"
- Module 3: "an arm of 1 DoF, which pushes the product inside the blank carton"
- Module 4: "On/off opposite arms to close the flaps"

Eventually, the system is subdivided in four modules and the functions are allocated to them. The Cartoner structure in Fig. 4.17 is a Block Definition Diagram (bdd) that shows the decomposition of the Cartoning machine into its modular blocks.

Next, for each module, it has been defined the Activity diagram and the related State machine diagram.

In the present work, we have considered "sunny day" conditions, in order to at most simplify the development of the modeling of the target manufacturing machine. Therefore we are focusing in the nominal system functionality, not considering fault detection, error handling, initialization and termination phases.

For the four modular blocks of the cartoning machine, at this point, further specifications are defined:

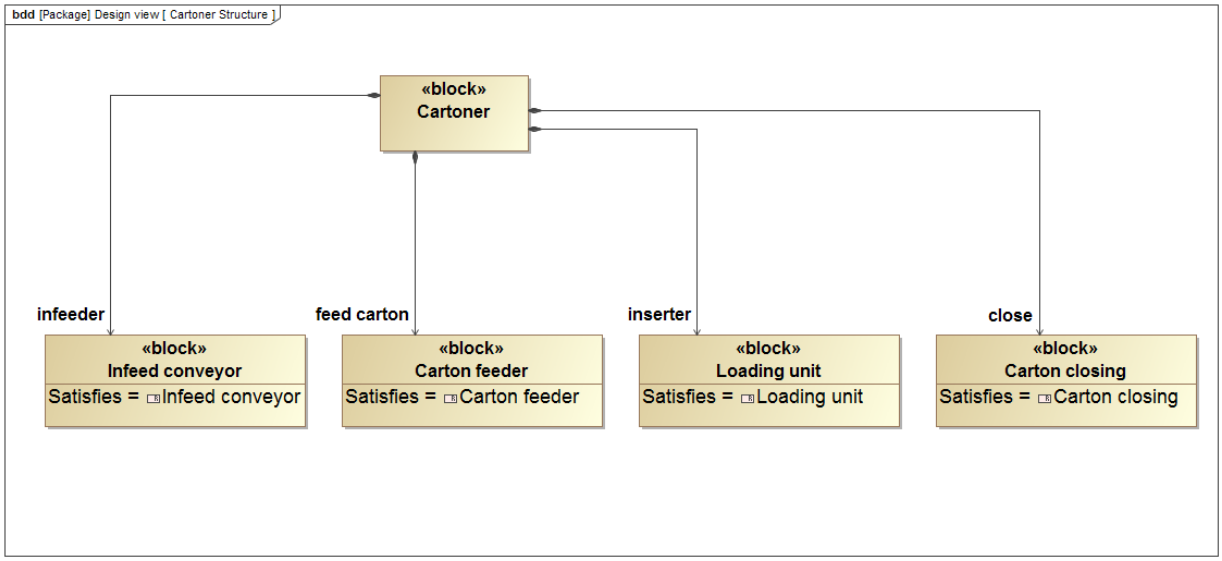

Figure 4.17: bdd [Package] Design view [Cartoner structure]

1. **Infeed conveyor**: it is the product pocket conveyor. The product pockets are conveyed along to the point from where the products will be loaded into a single carton.

2. **Carton feeder**: a vacuum fed arm pulls a flat carton from the carton magazine and places it in the pocket formed by chain lugs (see video [49]).

3. **Loading unit**: when the product in a bucket of the infeed conveyor, synchronized with a carton in the carton conveyor, reaches the loading station, a plunger pushes the article out of the bucket and into the open end of the carton.

4. **Carton closing**: flaps closing with glue.

Simulink/SimMechanics models

As reported in Fig. 3.7, Simulink models consists of SimMechanics model for the machinery and PID blocks for the relative controller.

For the Plant block, after identifying the rigid bodies and joints in the mechanism, each rigid body and their interfaces have been built via SimMechanics Second Generation, a tool of MathWorks MATLAB.

Then, in order to achieve the target tracking, in case, controllers of the single modules have been implemented with PID blocks.

For more efficient operations and energy savings, packaging machines have a number of sensors to monitor and control processes. Photoelectric sensors (or photoeyes, or photo-

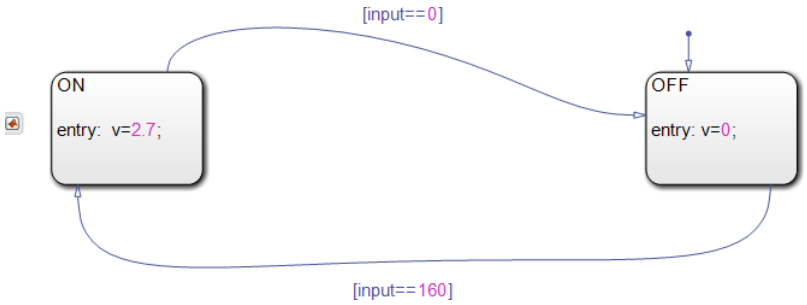

Figure 4.18: Simulink Model of Module 1, with Stateflow blocks

sensors) used in the present machine design are the convergent/proximity beam systems, which contains either the light source and the sensor in one housing. They use the package itself as the reflector and can be used to detect objects that cannot be easily detected by other photosensor systems [37].

Module 1 Module 1 is a conveyor system with two pulleys, and a straight, cleated belt. The system has an endless loop of carrying the conveyor belt, that rotates about the pulleys.

The output variable of the Infeed conveyor is the linear velocity of the conveyor itself. If the pockets on the conveyor have length 0.1 m, then the max speed of 160 crt/min corresponds to the linear speed v_{lin} = 16 m/min, i.e. v_{lin} = 0.27 m/s. With reference to rubber cleats T75 and belting in the Dunlop Conveyor Belting technical manual [51], we assume pulley diameters = 0.2 m, then rotational velocity of the pulleys is $\omega = \frac{v_{\text{lin}}}{D/2} = \frac{0.27}{0.1} = 2.7\,rad/s$.

We assume that upstream the products are discharged from the production line via an automatic infeed system, which is not part of our cartoning machine.

With this hypothesis for the process, it is not needed to directly control the infeed conveyor: product and/or carton absence will be checked in the next stations of the cartoner.

In Module 1 has not been employed neither SimMechanics blocks, nor PID controllers.

Module 1 has been modelled with Stateflow, and in Fig. 4.18 is shown the state machine of the Infeed conveyor.

It has one input and one output, respectively *input* and v: when *input*= 160 crt/min, then output v=2.7 rad/s, otherwise output v=0.

Input signals In the Simulink Module 1 model, the only input signal is the sum of two Step blocks, in order to have the system ON only at a certain interval time. When the sum is 0, then input=0 and the system is in state OFF.

Module 2 Module 2 is made up of a carton magazine, an arm which picks off blank cartons, and a chain conveyor, where cartons are positioned open.

The arm, provided with two vacuum suction cups, picks off a blank knocked-down carton at 45°, gripping the rear panel: that is, the panel which will be at the back of the pocket formed by the lugs on the conveyor. As the arm lands the carton on the conveyor at 0°, incoming lugs force it into its final squared-up shape, pushing against the side panel. The carton is now confined in the lug pocket by front and rear lugs. A rail underneath the open carton supports it vertically.

A pusher mechanism provides a constant pressure to keep the cartons at the front of the magazine. The pusher mechanism will keep uniform tension on the stack of cartons in the magazine whether full or nearly empty. This is critical, because too much or too little pressure can cause problems with picking off the carton. Small fingers, usually about 0.3 to 0.6 cm long at the front of the magazine, prevent the cartons from exiting and hold them in position for picking.

With reference to the cartoner max speed of 160 crt/min, 2.7 packages are produced every second, i.e. nearly one package every 0.33 s.

The arm is a stainless steel cylinder with radius=1 cm and length=30 cm.

Stainless steel is food-safe and is a proper material for operating in either Food& Beverage or Pharmaceutical plants.

The SimMechanics part employs a Revolute Joint, which has one rotational degree of freedom.

Control action is given by P controllers, i.e. K_P in the Position Loop and K_V in the Velocity Loop.

For coordination of operations and controls: one photosensor detecting the availability of folded cartons in the carton magazine and one photosensor detecting the presence of product.

Since they are different just in their position, it has been employed only the photosensor for the product, to keep the Simulink model simple.

In Simulink the photosensor should be a Proximity sensor of SimElectronics.

Unfortunately with the SimElectronics block Simulink cannot generate the C++ code file with TwinCAT target.

Therefore we represented the photosensor with a Sum block.

The Simulink model of Module 2 is shown in Fig. 4.19.

It has two inputs and two outputs:

- input qRef = reference arm position
- input To_Photosensor = the signal which in case notifies absence of products

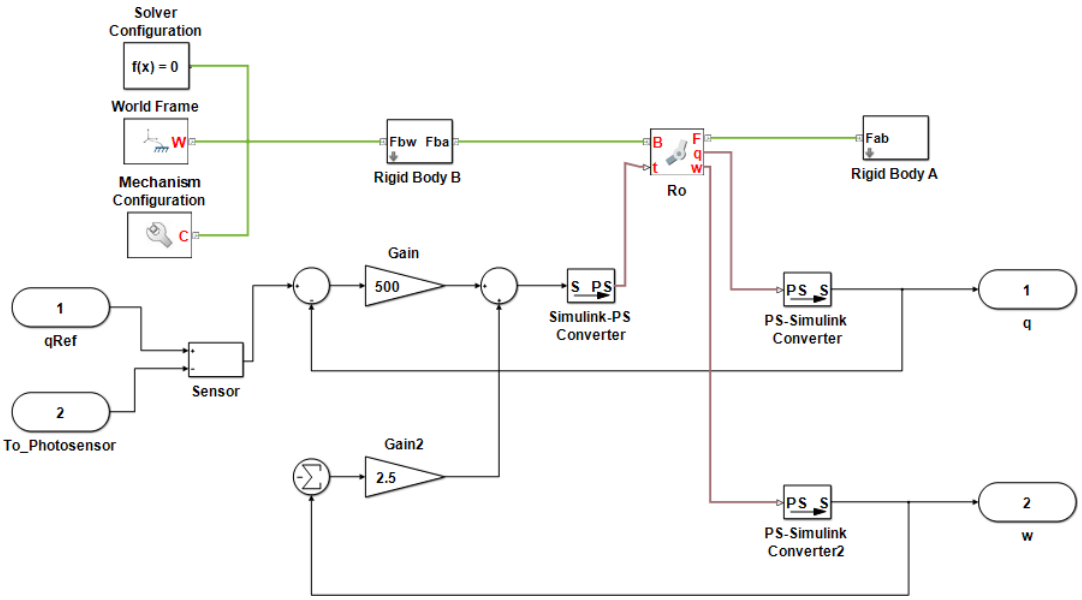

Figure 4.19: Simulink model of Module 2, which includes SimMechanics blocks

- output q = actual arm position
- output w = actual arm velocity

Input signals Inputs values for qRef and To_Photosensor are sinusoidal waves, hence they have the following expression:

$$y = Y \cdot sin(2\pi f \cdot t + \phi) + k, \tag{4.1}$$

where amplitude $Y = 0.3927\ rad = \frac{0.7854}{2}\ rad = 45°/2$, since the arm of the carton feeder goes from 0° to 45° and vice versa;

frequency f=3 Hz, since the maximum production speed gives three packages per second;

phase $\phi = -\frac{\pi}{2}$, so the arm of the carton feeder starts at 0°;

k=0.3927 rad, then the sinusoid has always positive values.

The aim is having the arm going to 45° (or 0.7854 rad) when a product is detected, in order to pick up the carton and release it at 0°.

Sinusoidal input is strongly recommended in packaging machines, since steep changes of velocity, like in triangular wave inputs, could diminish the life of machine, because of mechanical wear caused by applying big powers. Indeed, since $E = \frac{1}{2}mv^2$, a quick change in v causes high $\frac{dE}{dt} = P$, i.e. high powers, which soon or later bring to engine failure.

In the cartoner Simulink models sinusoidal inputs are provided via a Signal Builder Block, where the signals are custom waveforms. Data points of the waveforms are given by the following MATLAB expressions, according to equation 4.1:

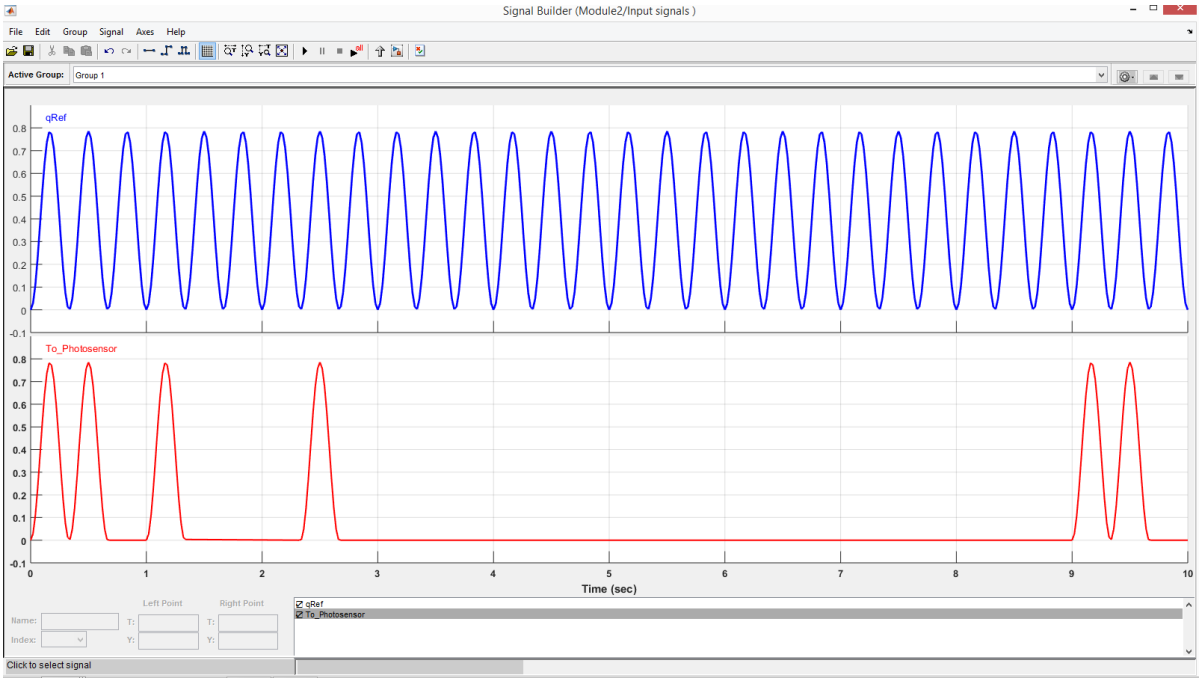

Figure 4.20: Signal Builder block with Input signals in Module 2

- Time values: t=[0:0.02:10]
- Y values: $y = 0.3927 * sin(6 * pi * t - (\frac{pi}{2})) + 0.3927$

With reference to the To_Photosensor signal: when the product is missing, the signal is zero and the carton feeder is stopped, conversely the signal is a sine wave.

In Fig. 4.20 in the Builder Signal block the two sinusoidal input signals of Module 2 are reported, i.e. qRef and To_Photosensor waveforms.

Module 3 The Loading unit consists of an actuated cylinder which pushes the product from the Infeed conveyor into the carton. It is assumed that the Infeed conveyor and the chain conveyor with cartons are aligned.

The actuated cylinder has made up of two solid blocks: an internal plunger with length=40 cm and an outer device 40 cm long which encloses the plunger and lets it get out longitudinally.

It is noteworthy that, in the Prismatic Joint, SimMechanics computes and applies the actuation force based on model dynamics, therefore in Module 3 there is no need to add PID blocks [52].

For coordination of operations and controls: two photosensors, in order to accomplish the loading only in the presence of both the product and the carton. Since they are

different just in their position, to keep the Simulink model simple it has been employed just the photosensor for the product.

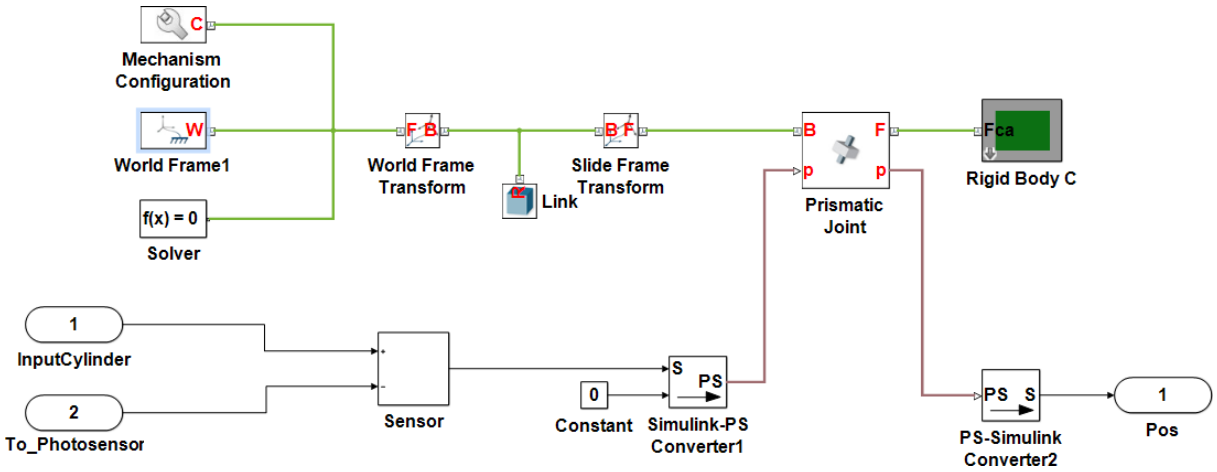

Figure 4.21: Simulink model of Module 3, which includes SimMechanics blocks. Output Pos is the actual position of the plunger

The Simulink model of Module 3 is shown in Fig. 4.21.

It has two inputs and one output:

- input InputCylinder = reference plunger position
- input To_Photosensor = the signal which in case notifies absence of products
- output Pos = plunger position

Input signals Concerning the inputs, also here we have sinusoidal waves and either signals follow equation 4.1:

Amplitude Y=0.15 m, since the plunger gets out of the base device, the clamping structure, of 0.30 m.

Frequency f=3 Hz, since the maximum production speed gives three packages per second.

Phase $\phi = -\frac{\pi}{2}$, so the plunger starts at position=0 m.

k=0.15 m, then the sinusoid has always positive values.

The aim is to have the piston reaching position up to 0.30 m, when a product is detected, in order to insert the product into the carton.

Also for the Loading station we have sinusoidal inputs, to avoid steep changes of velocity.

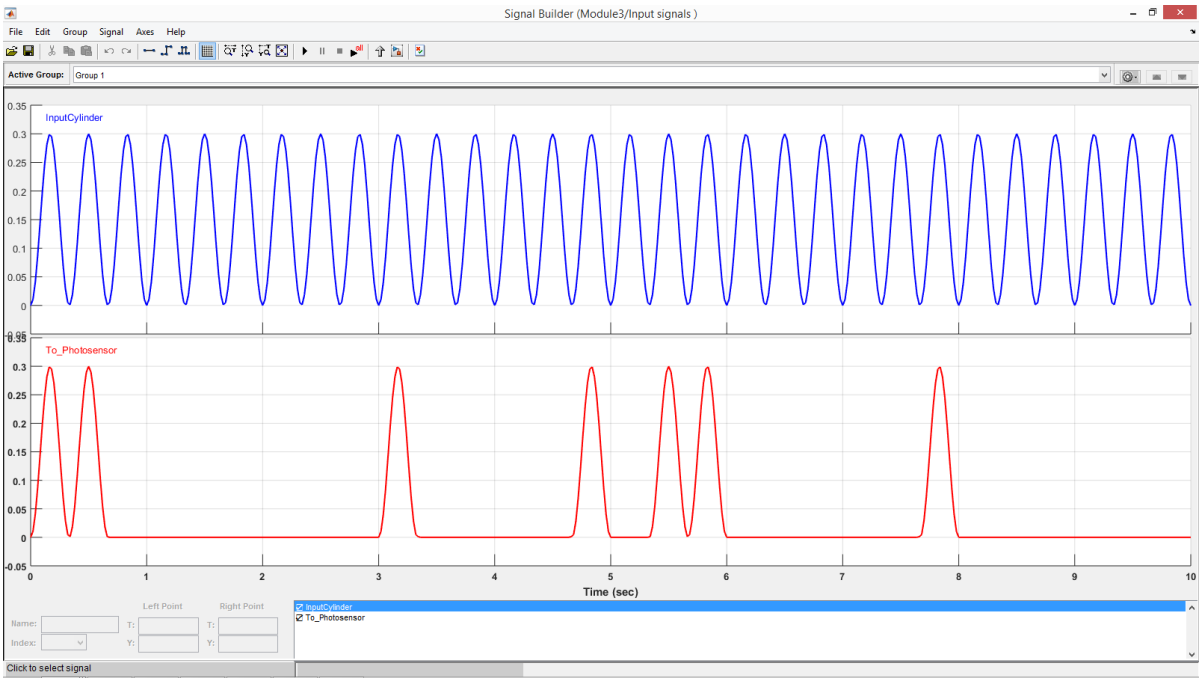

Figure 4.22: Signal Builder block with Input signals in Module 3

Input values are provided via a Signal Builder block, where the signals are custom waveforms. Data points of the waveforms are given by the following MATLAB expressions, according to equation 4.1:

- Time values: t=[0:0.02:10]
- Y values: $y = 0.15 * sin(6 * pi * t - (\frac{pi}{2})) + 0.15$

With reference to the To_Photosensor signal: when the product is missing, the signal is zero and the actuated cylinder is stopped, conversely the signal is a sine wave.

In Fig. 4.22 in the Builder Signal block are reported the two input sinusoidal signals of Module 3, i.e. InputCylinder and To_Photosensor waveforms.

Module 4 As the carton moves forward in the Carton closing station, on the relative side a fixed plough closes the leading minor flap, while a following rotating tucker closes the trailing minor flap. Then other two consecutive fixed ploughs close respectively the lower major flap and outer major flap. Before closing the outer major flap, a glue gun applies to the lower major flap an horizontal glue line. The major flap closing plough, one for each side, folds the outer major flap down onto the glue and holds the flaps closed.

For this closing schematic, see Fig. 4.23, where the glue head is omitted. We have resolved to neglect the glue head because it is trivial and, in turn, the closing schematic

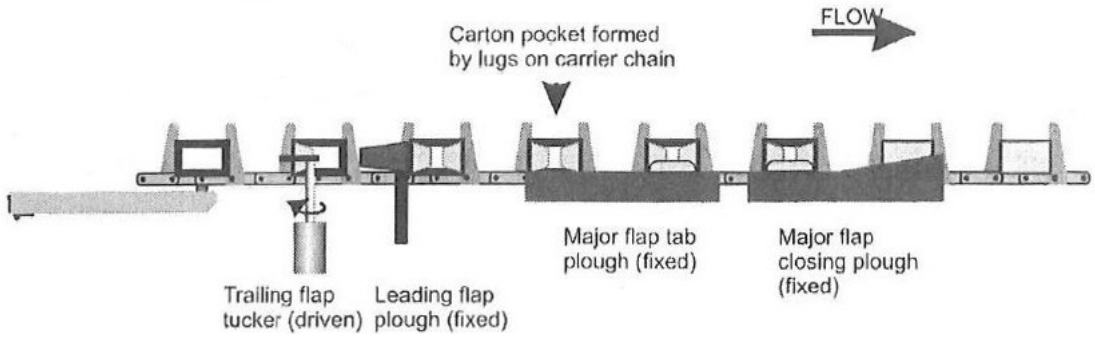

Figure 4.23: Carton closing schematic (source: Institute of Packaging Professionals)

can be considered both for pre-glued and normal carton. Finally, for sealing the carton, a pusher (Actuated cylinder), one for each side, compresses the flaps for good adhesion.

Since the glue guns have been omitted, the only electrically driven elements are then the following sub-modules:

- Trailing flap tuckers
- Actuated cylinders

Eventually the Module 4 consists of four sub-modules, i.e. one Trailing flap tucker and one Actuated cylinder for either sides of the chain conveyor.

For coordination of operations and controls: one photosensor for both the Actuated cylinders.

Trailing flap tuckers have not sensors, because they are supposed to be always running.

Fig. 4.24 is an image of Mechanics Explorer from Module 4 Simulink simulation. The distance between the two Trailing flap tuckers and the Actuated cylinders is 1 meter.

The Simulink model of Module 4 is shown in Fig. 4.25.

The sub-modules Actuated cylinders are specular and correspond to Module 3, the Loading unit, and have the same input signals. Therefore, with reference to the Simulink Module 4, hereafter it will be described just the sub-module Trailing flap tucker.

Also the two Trailing flap tuckers are specular. One Trailing flap tucker runs clockwise, while the Trailing flap tucker on the other side of the conveyor runs counter-clockwise, so both of them can close the relative trailing minor flap of the package.

Each Trailing flap tucker consists of five rigid bodies and a Revolute Joint.

Two rigid bodies forms the rotating disk about its vertical axis above a cylinder. A base supports the cylinder.

The disk makes three rotations per minute, as the production capacity requires.

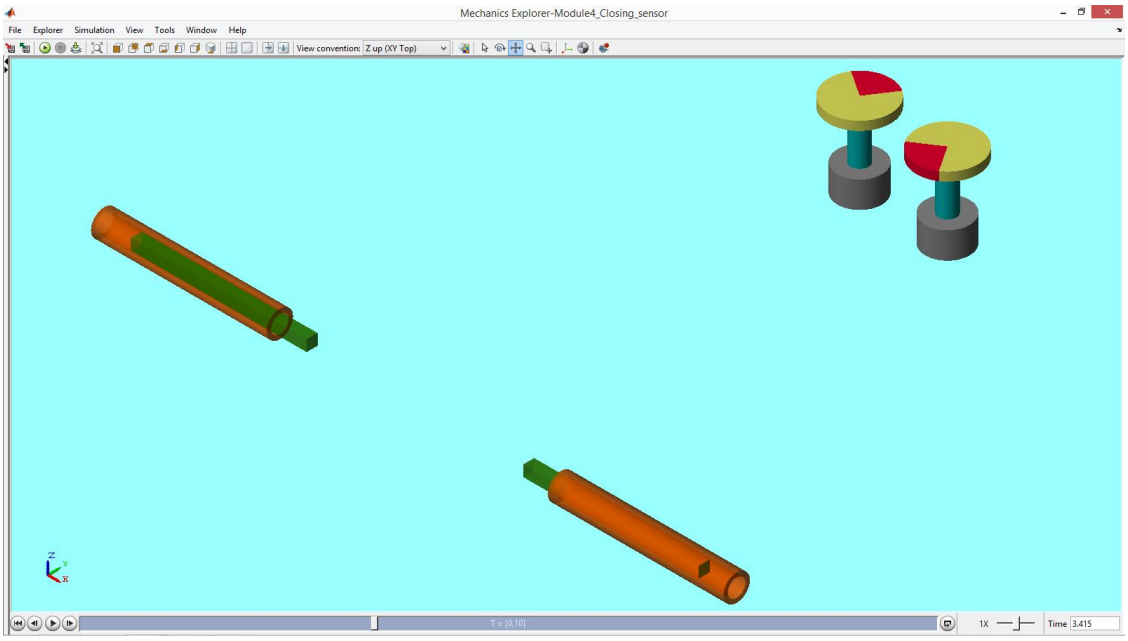

Figure 4.24: An image of Mechanics Explorer from Module 4 Simulink simulation

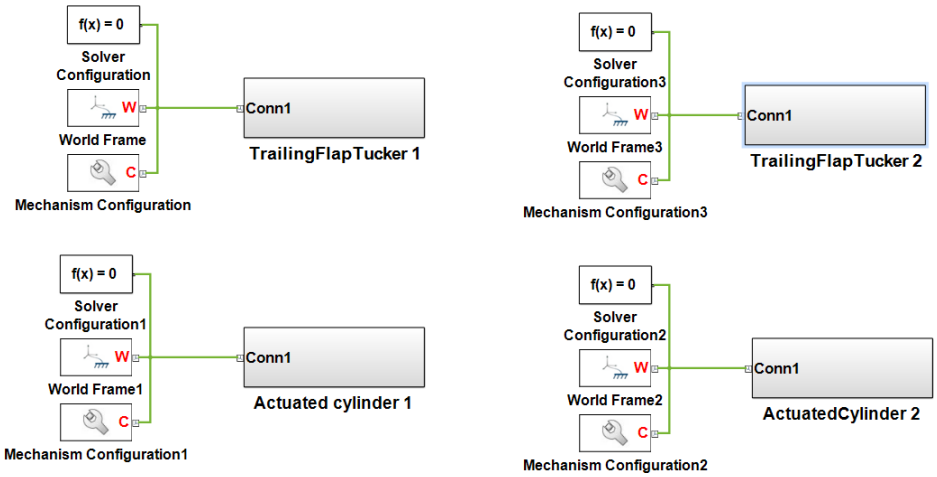

Figure 4.25: Simulink model of Module 4, which is formed by four Simulink sub-modules

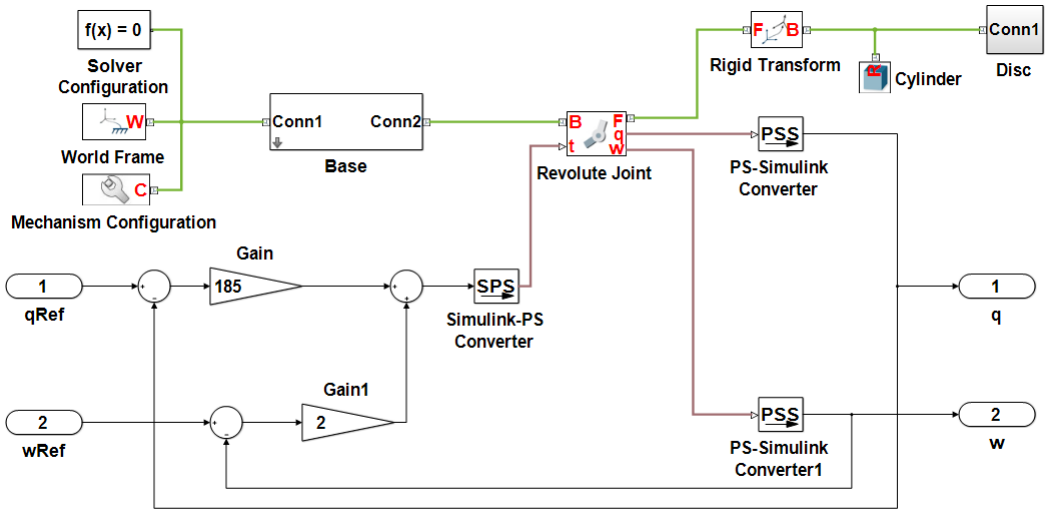

Figure 4.26: Simulink model of Trailing flap tucker

The two rigid bodies of the disk are respectively the red minor and the yellow major circular sector. This distinction has been deemed useful to see the disk spinning in the Simulink simulation.

The remaining three rigid bodies in the Trailing flap tucker are the cylinder which supports the disk and a base which supports all the other bodies and gives sturdiness to the tucker. See Fig. 4.26.

Control action is given by P controllers, i.e. K_P in the Position Loop and K_V in the Velocity Loop.

Trailing flap tucker has two inputs and two outputs:

- input qRef = reference disk position
- input wRef = reference disk velocity
- output q = actual disk position
- output w = actual disk velocity

Input signals Relating the input signals of the two Actuated cylinders, refer to the input signals described in Simulink Module 3, because they are employed even in Module 4, with the same values.

Input qRef values are applied to the Trailing flap tucker via a Repeating Sequence block, as follows:

- Time values: [0 0.167 0.333 0.5 0.667 0.833 1]
- Output values: [0 -pi -2*pi -3*pi -4*pi -5*pi -6*pi]

Input wRef values are applied to the Trailing flap tucker via a Repeating Sequence block, as follows:

- Time values: [1 0]
- Output values: [0 0]

  In t=1+n s, with n=1..9, v=0, because in these instants the rotation reference, being in position $-(6 \cdot n) \cdot \pi$, does not need to move for reaching position 0 rad.

This means that with the above inputs, the disk performs three complete turns clockwise.

Obviously for the other Trailing flap tucker the inputs have values for counter-clockwise turns.

Configuring Simulink parameters The Solver and the Code generation parameters have been set as in the Left JSU project, except for the simulation time, which in the Cartoning machine is 10 s.

Beckhoff TwinCAT projects

In Module 2, 3, and 4 signals inputs are provided via .csv files which have two columns, one for the time and the other for the relative values.

In the upcoming paragraphs, steps to implement HIL simulations for the Modules of the cartoning machine in TwinCAT will be described, focusing on MOTION and PLC, the only configuration parts which differ.

Module 1 Module 1 does not need motion control, since input has just two values. We can handle these two values in the PLC code.

PLC In Module 1 the PLC module consists of three Program Organization Units (POUs), the MAIN program and two functional blocks, i.e. FB_Mod1_InfeedConveyorB_TCT and FBTcMatSimObject. The two functional blocks implements a set of methods functionalities and the state machine handling (see Fig. 4.27).

This PLC structure has one more POU, compared to the Left JSU PLC MAIN program, due to an added FBTcMatSimObject in the more recent version of Beckhoff TwinCAT software.

Module 2

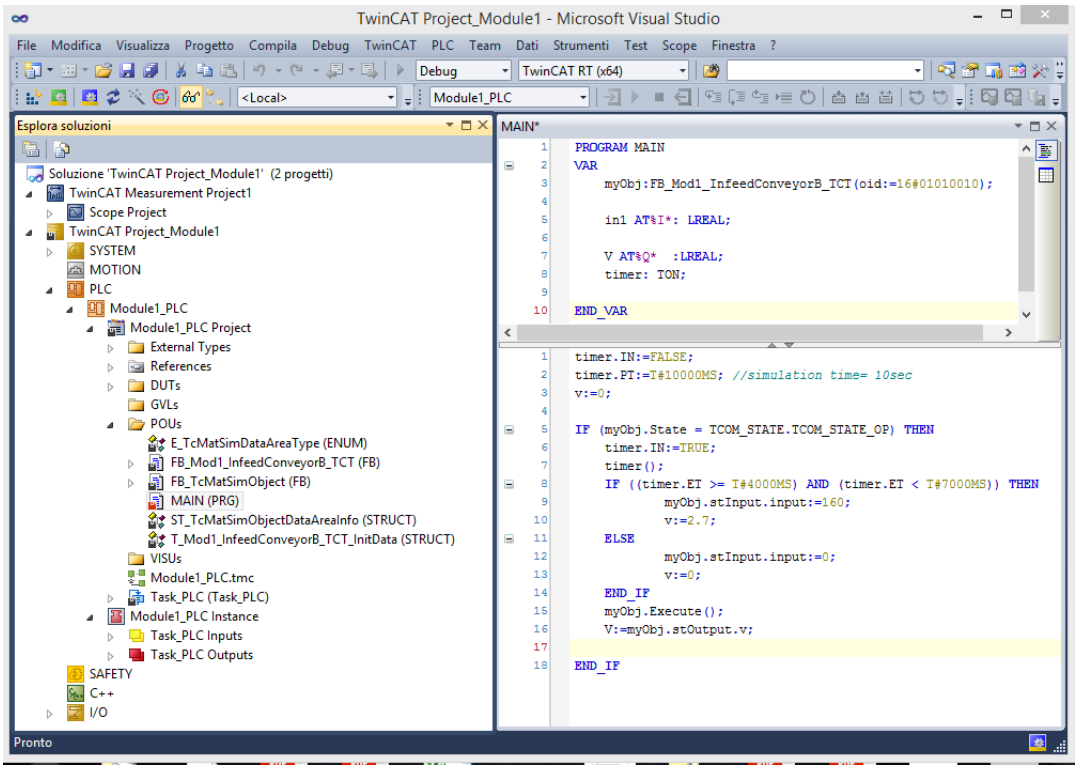

Figure 4.27: Module 1 - PLC implementation

MOTION Module 2 has two input .csv files, i.e. qRef and To_Photoeye, with the same values of those ones in the corresponding Simulink model. Their values have been prepared in Microsoft Excel, in particular qRef and To_Photosensor according to the following equation:

$$y(t) = 0.3927 \cdot sin(2\pi \cdot 3 \cdot t - \frac{\pi}{2}) + 0.3927.$$

The user's interface to the cam design editor is graphic, as it can be seen in Fig. 4.28.

PLC PLC module consists of three Program Organization Units (POUs), the MAIN program and two functional blocks, i.e. FB_Mod2_020915_TCT and FBTcMatSimObject. The two functional blocks implement a set of methods and the state machine handling (see Fig. 4.29). An Action, i.e. an additional POU implementation which is assigned to the MAIN program, handles the motion control.

Module 3

MOTION Also Module 3 has two input .csv files, i.e. InputCylinder and To_Photoeye, with the same meaning of those ones in the corresponding Simulink model. Their values

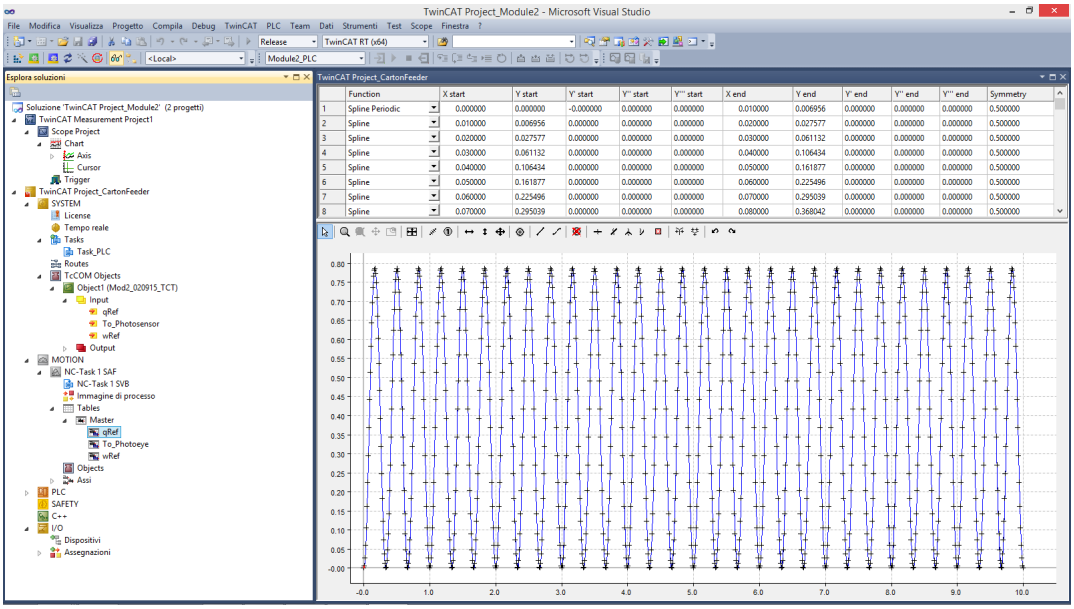

Figure 4.28: Module 2 - Visualization of input qRef in TwinCAT CAM Design Tool

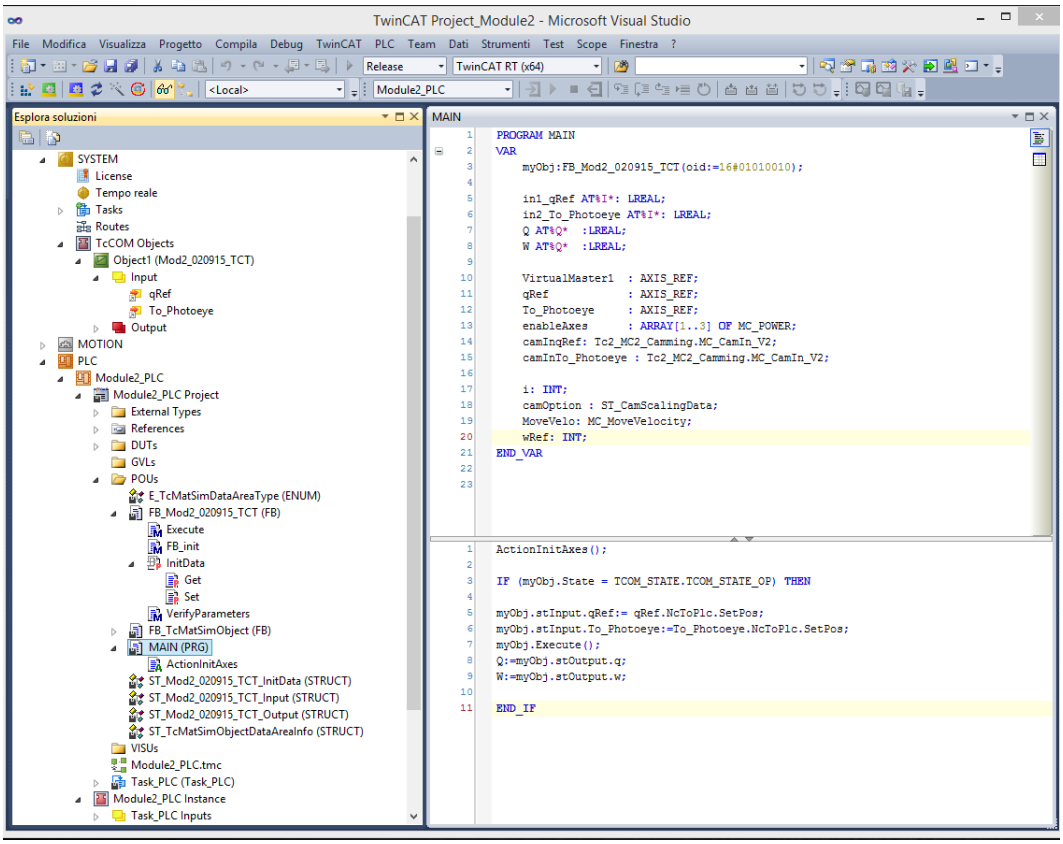

Figure 4.29: Module 2 - PLC implementation

have been prepared in Microsoft Excel according the following equation:

$$y(t) = 0.15 \cdot sin(2\pi \cdot 3 \cdot t - \frac{\pi}{2}) + 0.15.$$

PLC The PLC module consists of three Program Organization Units (POUs), the MAIN program and two functional blocks, i.e. FB_Mod3_ActCyl_Sensor and FBTcMatSimObject. The two functional blocks implement a set of methods and the state machine handling, as for Module 2 PLC. Also here an Action, as for Module 2 PLC, handles the motion control.

Module 4 Module 4 is formed by four different sub-modules: two Trailing flap tuckers and two Actuated cylinders. Thus in SYSTEM there are four distinct objects, each one generated in MATLAB/Simulink by means of Model Referencing.

MOTION In Motion there are as many cams as the number of position inputs of the devices.

With reference to the Trailing flap tuckers, they do not need a cam for the velocity input wRef, because velocity values can be derived directly from the position input at the same time instants. Therefore wRef inputs are supplied connecting them to the corresponding qRef axes as *SetVelo*.

PLC This Module has just one PLC for all the four sub-modules.

Obviously the code for the relative task is more complex than it was in the previous modules, but the principle of the program MAIN instantiating functional blocks of the objects is still the same. As in precedent modules, program MAIN has Action files in order to handle motion control of all the four devices.

4.3.2 Experimental validation for the Carton feeder

In the Module 2 of the Cartoning machine, a blank carton is pulled out of the carton magazine and positioned open on a chain conveyor.

Fig. 4.30 is a snapshot of the scope for the Simulink simulation of Module 2, while Fig. 4.31 is a snapshot of the scope for the HIL simulation of the same module.

Both the two images show the trend of respectively the reference signal qRef and output q, which represent the position of the arm of the Carton Feeder.

Output q signal is the blue graph in Simulink scope and in HIL scope as well.

In either Simulink and in HIL simulation, inputs are sinusoidal waves, with phase –90°, so they begins from zero. When a product is detected, output q (in blue) traces a

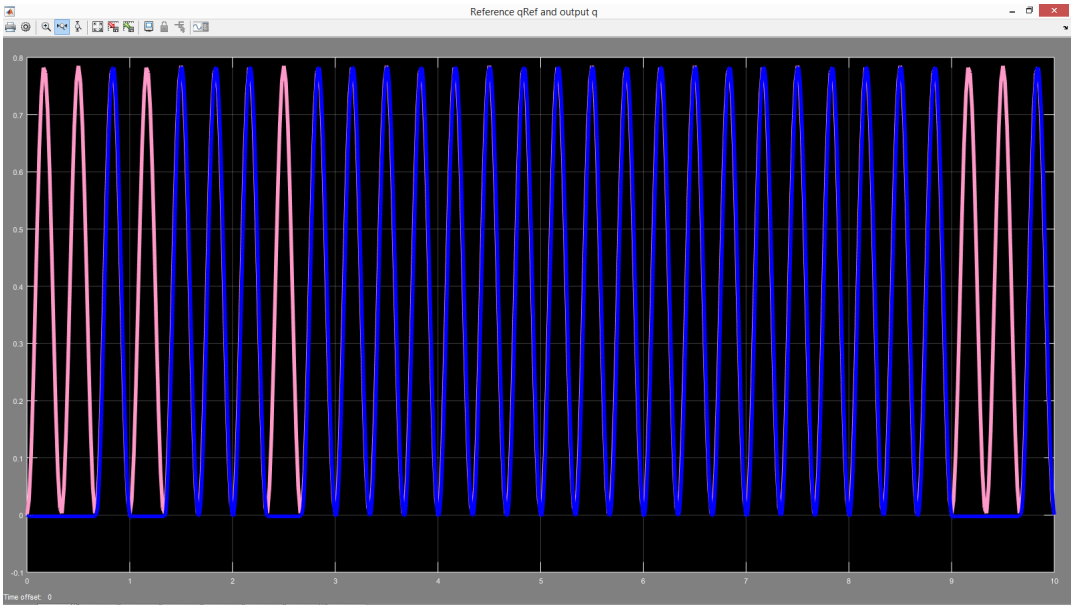

Figure 4.30: Module 2 - Reference qRef and output q in the Simulink simulation

sinusoid and the arm goes to 45°(or 0.7854 rad), in order to pick up the carton and release it at 0°. Output q is zero when the relative photosensor do not detect the product.

The results of the Simulink simulation and of the HIL simulation are satisfactory, because both of them reproduce correctly the position of the arm of the Carton feeder, i.e. output q.

Module 2 HIL project provides a proof of concept of reusability for the HIL scheme validated in the Left JSU HIL project.

Moreover the brand new TwinCAT/HIL scheme built on just on laptop is validated even at modular HIL level [4].

4.3.3 Experimental validation for the Carton closing

In the Cartoning machine, Module 4 refers to the Carton closing process, in which open flaps of the carton are closed with glue.

Module 4 is a composition of two Trailing flap tuckers and two blocks of Module 3, i.e. two modules of an already tested modular block, i.e. the Actuated cylinder.

In fact, Module 3, that is the Loading unit in the Cartoning machine, has the same features of the Actuated cylinder, therefore it is re-used.

Fig. 4.32 and Fig. 4.33 refer to Simulink simulation of respectively one of the two Trailing flap tuckers and one of the two Actuated cylinders of Module 4.

Both the two images show the trend of respectively the reference position and the actual position.

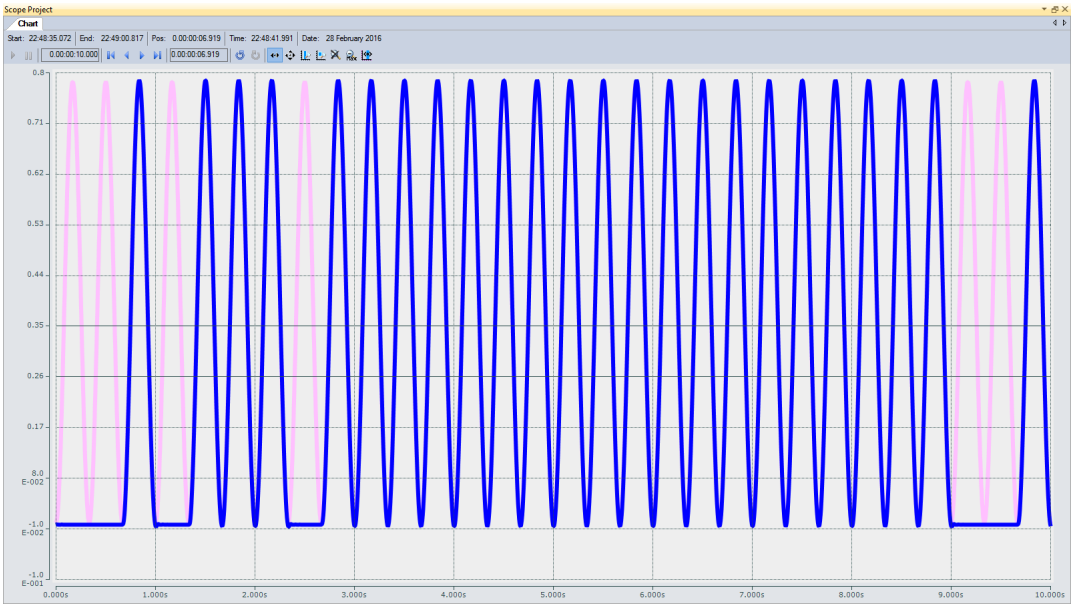

Figure 4.31: Module 2 - Reference qRef and output q in the HIL simulation

In particular the tucker accomplishes three loops (until -6π) per second, assuming that it is the one which spins clockwise. That is because the maximum capacity production requires 160 crt/min, i.e. nearly 3 packages per second. The Trailing flap tuckers are supposed to be always running.

With reference to Fig. 4.33, representing the reference signal and output Pos of an Actuated cylinder, signals are sinusoidal waves, with phase $-\frac{\pi}{2}$, so it begins from zero. Sinusoidal input are strongly recommended in packaging machines, since steep changes of velocity could diminish the life of the machine, because of mechanical wear caused by applying big powers.

In conformity with the requirements of the Cartoner project, during packaging process Trailing flap tuckers are always on, while Actuated cylinders work only when a product is detected by the photosensor, conversely relative output signal is zero.

In Fig. 4.32 the output varies from the reference signal only in t=0, but it is evident that the correct value is immediately recovered. In the successive instants of the simulation period, the output q is correct.

Reference and output signal in Fig. 4.33 overlap perfectly, except, obviously, when a product is missing, and then the plungers are still. In these cases, only reference signal, in pink, is on, while output Pos, in green, is at position zero.

In Module 4, while Simulink simulation concerns the whole module, with TwinCAT two different kind of HIL simulations have been implemented, one for each single sub-module, i.e. the Trailing flap tucker and the Actuated cylinder; the other one via an overall

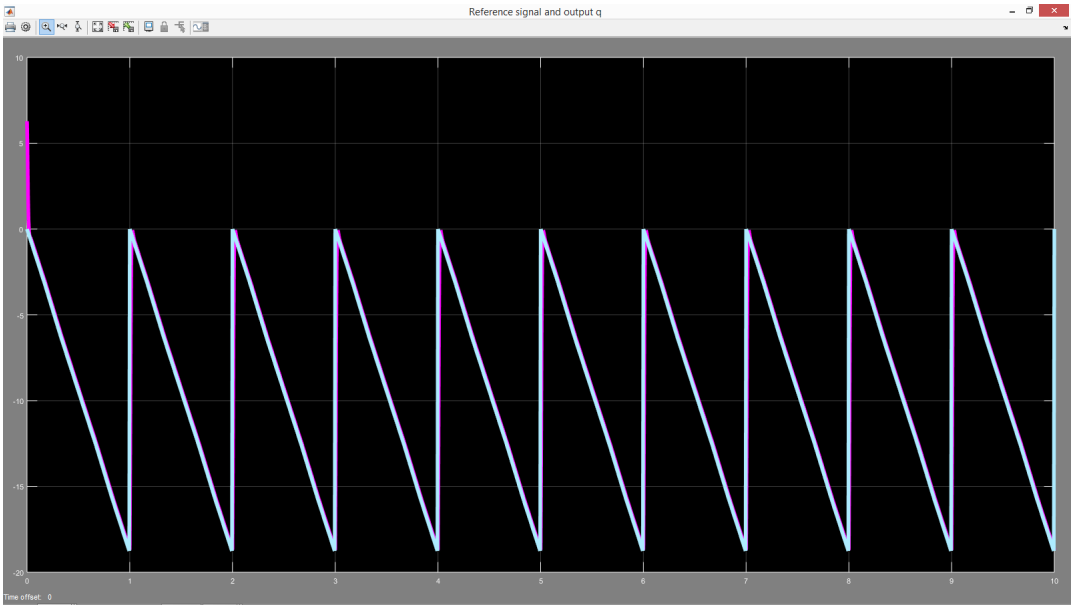

Figure 4.32: Module 4 - Reference position qRef and actual position q of one of the two Trailing flap tuckers, in the Simulink simulation

model extended to the four sub-modules and a PLC. Even in this last HIL simulation, a unique CPU is employed, both for the host computer, which executes the Cartoning machine plant model in real-time, and the control system. Fig. 4.34 is a chart of the trends of reference and output signals of the four sub-modules of Module 4, in the HIL simulation.

The results of the Simulink simulation and of the HIL simulation are satisfactory, since position signals of the devices in Module 4 are correctly reproduced.

The successful Module 4 HIL simulation brings the following key strategic advances in mechatronic design:

1. Module 4 HIL project provides a proof of concept of reusability and reconfigurability by successfully assembling an already available modular block [5].

2. It provides the feasibility of integrating many Simulink modular models in a composite model, running on one PLC and one CPU [6].

3. It successfully implements the whole MBSE design and virtual testing of the workflow proposed in the previous chapter [5] [6].

4. It is a virtual commissioning inside the design Cartoning machine project, since the HIL simulation implements the V&V phase of the right side of the V diagram [5] [6].

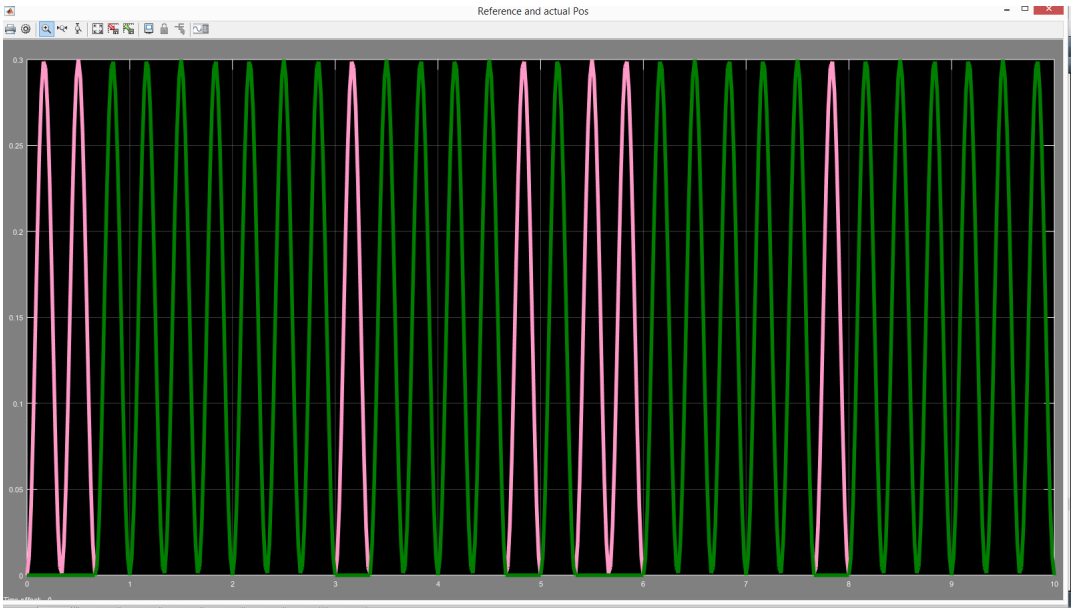

Figure 4.33: Module 4 - Reference position InputCylinder and actual position Pos of one of the two Actuated cylinders, in the Simulink simulation

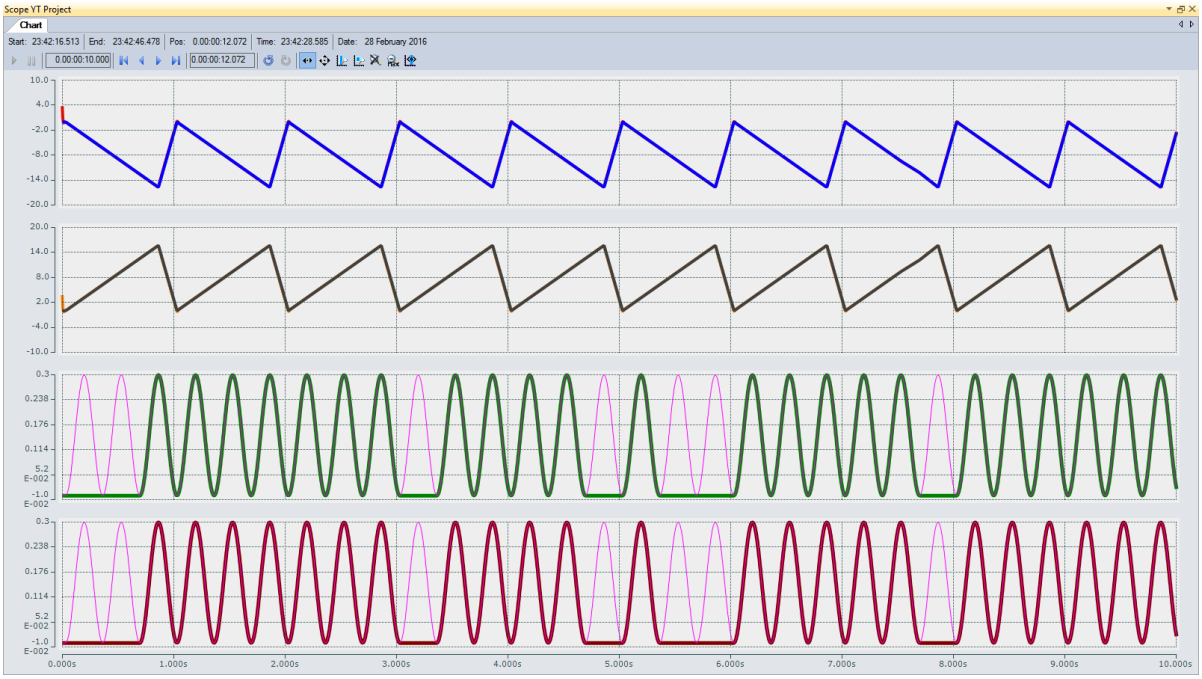

Figure 4.34: Module 4 - Reference and actual outputs of the four sub-modules in the HIL simulation

4.4 Discussion

Validation of theoretical findings is a critical task when developing new strategies and methodologies. Providing proofs of concept and possible applications of the theoretical parts allow to understand practically the advantages and the usefulness of what has been proposed. The control strategies and the software architecture presented in Chapters 2 and 3 have been implemented on Jaw System Unit HIL project and on a Cartoning machine design project.

The results of related experiments have been discussed in this chapter.

The first case study addresses the validation of a brand new virtual HIL scheme, developed in Beckhoff TwinCAT environment, with integration of PLC, Motion Control, and Simulink/SimMechanics model on a unique CPU, which is employed both for the real time execution of the plant model and the control system, in an ordinary personal computer, without any other hardware tool. Moreover the Left JSU HIL project confirms the expansion of possibilities for TwinCAT development platform, also for implementing simulation capabilities.

The second case study has been discussed more intensively, since it implements the whole software architecture proposed in Chapter 3. Following an MBSE approach, a cartoning machine has been designed from scratch according to a modular system development. Experimental results of two particular modular structures of the the cartoning machine project have been reported, namely the Carton feeder and the Carton closing.

The positive results of both the two use cases bring key strategic advances in mechatronic design.

While Carton feeder results validate the virtual HIL scheme built on just on laptop even at modular level, the results of the Carton closing provide a proof of concept of reusability for the HIL scheme validated in the Left JSU HIL project, and the feasibility of integrating many Simulink modular models in a composite one model, running on one PLC and one CPU.

Furthermore, Carton closing module successfully implements reconfigurability of the proposed software methodology. In fact, it is a composite module which includes, as a sub-module, another module of the Cartoning machine.

Carton closing HIL simulation is also a virtual commissioning inside the design Cartoning machine project, since the HIL simulation implements the Verification & Validation phase of the right side of the V diagram.

Finding problems before you commit to hardware leads to huge cost savings, including benefits like being able to optimize global system performance, and shortening design iterations, reducing overall development time.

Chapter 5

Concluding remarks

Literature reviews confirm that hardware-in-the-loop is currently under consideration by several researchers for providing support to the verification and validation of mechatronic systems.

Since HIL simulation employs a real controller, it is more efficient than software-in-the-loop (SIL) simulation, which is not a real-time test, or model-in-the-loop (MIL) simulation, which is real-time, but with both controller and plant simulated.

This thesis dealt with improving packaging machine development via HIL simulations for virtual commissioning of control software. Packaging machines are mechatronic systems, which, in the Industry 4.0 perspectives, will become more and more complex. Competitiveness imposes to machine builders short time-to-market and exigencies of increased functionality, reliability, and safety. The answer proposed in this thesis is HIL simulation of machines before they are even built, allowing possible weaknesses to be detected and corrected early on. During HIL simulation we have real-time simulating machine behaviour, with the options to change system parameters and test the machine limits. Damaging the machine is not possible, and the influence of external forces can be depicted.

Moreover the interactions of heterogeneous control hard and software plays a key role in enabling mechatronic production system to become flexible and agile systems.

Virtual commissioning, which in this thesis is carried out through HIL tests, enables control software engineering to both take over the initiative in system design and to perform simulations earlier in the design process of production equipment.

The proposed HIL strategy employs a novel scheme with integration of PLC, Motion Control, and Simulink/SimMechanics models on a unique CPU, both for the real time execution of a modular plant model and the control system, on just one laptop.

To this aim, a dedicated software architecture has been implemented. The development workflow follows the MBSE paradigm, which is a must in complex mechatronic

systems design.

The brand new HIL environment brings huge opportunities of overall quality improvements and restrained development costs.

It already implements key Industry 4.0 principles, like modularity, virtualization and real-time capability.

New technologies like Virtual Reality, Augmented Reality, and Artificial Intelligence are promising elements for future advances in the HIL test system. Actually they could provide a complete bench top simulation and may make the HIL simulation an absolute replacement for a real packaging machine for control system validation and verification.

Bibliography

[1] M. Hermann, T. Pentek, and B. Otto, "**Design principles for Industrie 4.0 scenarios: a literature review**," *Technische Universität Dortmund, Dortmund*, 2015.

[2] "Information on **Industry 4.0**," Accessed date 11-February-2016. [Online]. Available: https://www.rolandberger.com/media/pdf/Roland_Berger_TAB_Industry_4_0_20140403.pdf

[3] "Information on **Mechatronics' Evolution Impacts Use of Modeling and Simulation**," Accessed date 19-February-2016. [Online]. Available: http://www.designnews.com/document.asp?doc_id=229574

[4] R. Tessari and C. Fantuzzi, "**Modularization of a packaging machine encompassing Hardware-In-the-Loop Simulation**," in *2015 International Conference on Simulation, Modelling and Mathematical Statistics, Chiang Mai, Thailand – SMMS 2015*. DEStech Publications, November 2015, pp. 1–5, in press.

[5] R. Tessari and C. Fantuzzi, "**Design of a packaging machine encompassing modular Hardware-In-the-Loop Simulation**," in *2015 International Conference on Applied Mechanics and Mechatronics Engineering, Bangkok, Thailand – AMME 2015*. DEStech Publications, October 2015, pp. 1–5, in press.

[6] R. Tessari and C. Fantuzzi, "**Design of a packaging machine and Virtual Commissioning via Modular Hardware-in-the-loop Simulations**," in *18TH Mediterranean Electrotechnical Conference, Limassol, Cyprus – MELECON 2016*. IEEE, April 2016, pp. 1–5, in press.

[7] "Information on **Fabbrica Intelligente**," Accessed date 29-February-2016. [Online]. Available: http://www.fabbricaintelligente.it/progetti/

[8] R. Chen, L. Mi, and W. Tan, "**A New Hardware-In-The-Loop Test System for Electronic Control Unit of Dual-Clutch Transmission Vehicle**," *Advanced Materials Research*, vol. 490, pp. 13–18, March 2012.

[9] R. S. Kalawsky, J. O'Brien, S. Chong, C. Wong, H. Jia, H. Pan, and P. R. Moore, "**Bridging the Gaps in a Model-Based System Engineering Workflow by Encompassing Hardware-in-the-Loop Simulation**," vol. 7, pp. 593–605, December 2013.

[10] H. Wenbo, Z. Yinhui, J. Zhenyu, and Z. Weihua, "**Hardware-in-the-Loop Simulation and Flight Experimentation of the Control System on a Small Solid Rocket**," *Applied Mechanics and Materials*, vol. 232, pp. 482–486, November 2012.

[11] M. Verbrugge, D. Frisch, T. Weber, A. Bekaryan, and P. Liu, ***Examination of State Estimators for Electrochemical Energy Storage Devices by Means of A Hardware-in-the-Loop System***. Nova Science Publishers, 2009, vol. Electric Vehicles: Technology, Research and Development, ch. 4, pp. 87–103.

[12] A. M. El-Nagar and M. El-Bardini, "**Interval type-2 fuzzy neural network controller for a multivariable anesthesia system based on a hardware-in-the-loop simulation**," vol. 61, pp. 1–10, May 2014.

[13] D. Titterton and J. L. Weston, ***Strapdown Inertial Navigation Technology, 2nd Edition***, ser. IEE Radar, Sonar, Navigation and Avionics Series. IET, 2004, ch. 8, ISBN: 978-0863413582.

[14] B. Allotta, R. Conti, E. Meli, L. Pugi, and A. Ridolfi, "**Development of a HIL railway roller rig model for the traction and braking testing activities under degraded adhesion conditions**," *International Journal of Non-Linear Mechanics*, vol. 57, pp. 50–64, December 2013.

[15] J. Burbank, W. Kasch, and J. Ward, ***An Introduction to Network Modeling and Simulation for the Practicing Engineer***. Wiley-IEEE Press, 2011, ch. 6: Hardware-in-the-Loop Simulations, pp. 114–142.

[16] S. C. Park and M. Chang, "**Hardware-in-the-loop simulation for a production system**," vol. 50, pp. 2321–2330, April 2012.

[17] C.-R. Rad, V. Maties, O. Hancu, and C. Lapusan, "**Hardware-in-the-Loop (HIL) Simulation used for Testing Actuation System of a 2-DOF Parallel Robot**," *Applied Mechanics and Materials*, vol. 162, pp. 334–343, March 2012.

[18] Z. M. Zhong, X. P. Chen, G. L. Kong, and X. B. Chen, "**Hardware-in-the-loop Test and Failure Mode Simulation for AMT**," *Applied Mechanics and Materials*, vol. 188, pp. 292–299, June 2012.

[19] C. Fantuzzi, R. Panciroli, and M. Gargiulo, "**Hardware in the loop simulation for Distributed Automation Systems**," in *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*. IEEE, September 2012, pp. 1–6.

[20] "Information on **Typical HIL simulation setup**," Accessed date 02-February-2016. [Online]. Available: http://it.mathworks.com/help/physmod/simscape/ug/what-is-hardware-in-the-loop-simulation.html

[21] O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, "**Development of a driver information and warning system with vehicle hardware-in-the-loop simulations**," *Mechatronics,*, vol. 19, no. 7, pp. 1091–1104, October 2009.

[22] J. Keränen and T. Räty, "**Model-based testing of embedded systems in hardware in the loop environment**," *IET Software*, vol. 6, p. 364 – 376, August 2012.

[23] J. H. Li, C. X. Song, and L. Q. Jin, "**The Benchmarking of ESP Hardware-in-the-Loop Test Bench With Data From Real Vehicle**," *Advanced Materials Research*, vol. 482-484, pp. 478–482, February 2012.

[24] B. Alecsa, M. Cirstea, and A. Onea, "**Simulink Modeling and Design of an Efficient Hardware-Constrained FPGA-Based PMSM Speed Controller**," vol. 8, pp. 554–562, August 2012.

[25] "**B&R Automation**," Accessed date 02-February-2016. [Online]. Available: http://www.br-automation.com/en/perfection-in-automation/

[26] "**Beckhoff**," Accessed date 02-February-2016. [Online]. Available: http://www.beckhoff.com/

[27] "**Siemens Automation**," Accessed date 02-February-2016. [Online]. Available: http://www.siemens.com/entry/cc/en/

[28] "Information on **FiO board**," Accessed date 02-February-2016. [Online]. Available: www.aimagin.com

[29] "Information on **MATLAB/Simulink**," Accessed date 02-February-2016. [Online]. Available: www.mathworks.com

[30] "Information on **ControlDesk Next Generation**," Accessed date 02-February-2016. [Online]. Available: https://www.dspace.com/en/inc/home/products/sw/experimentandvisualization/controldesk.cfm

[31] "Information on **LMS Amesim**," Accessed date 02-February-2016. [Online]. Available: http://www.plm.automation.siemens.com/it_it/products/lms/imagine-lab/amesim/platform/real-time-simulation.shtml

[32] F. P. Queiroz, F. J. Gomes, L. P. De Freitas, and V. A. Gama, "**Development of a Foss-Based Hardware-in-the-loop Platform for Control Engineering Education**," *Journal of Control, Automation and Electrical Systems*, vol. 24, no. 3, pp. 244–252, June 2013.

[33] "Information on **OPC technology**," Accessed date 03-February-2016. [Online]. Available: http://www.opcfoundation.org/

[34] "Information on **COM technology**," Accessed date 03-February-2016. [Online]. Available: http://www.microsoft.com/COM/

[35] J. A. Estefan *et al.*, "**Survey of model-based systems engineering (MBSE) methodologies**," *Incose MBSE Focus Group*, vol. 25, no. 8, 2007.

[36] G. Reinhart and G. Wünsch, "**Economic application of virtual commissioning to mechatronic production systems**," vol. 1, pp. 371–379, December 2007.

[37] J. R. Henry, ***Packaging Machinery Handbook***. Createspace, 2012.

[38] D. Borghi and C. Fantuzzi, ***Control of Mechatronic systems***. UniMoRe, 2012.

[39] "Information on **Tetra Pak**," Accessed date 19-February-2016. [Online]. Available: http://www.tetrapak.com/

[40] "Information on **Massive software package**," Accessed date 03-February-2016. [Online]. Available: http://www.massivesoftware.com/engineering.html

[41] S. Friedenthal, A. Moore, and R. Steiner, ***A practical guide to SysML: the systems modeling language***. Morgan Kaufmann, 2014.

[42] "Webinar: **'We chose MBSE: what's next? Best practises in systems modeling'**," No Magic, Inc., Accessed date 04-February-2016. [Online]. Available: https://www.youtube.com/watch?v=KMwI44aK8f0

[43] G. Barbieri, C. Fantuzzi, and R. Borsari, "**A model-based design methodology for the development of mechatronic systems**," *Mechatronics*, vol. 24, no. 7, pp. 833–843, 2014.

[44] C. McLean and S. Leong, "**The expanding role of simulation in future manufacturing**," in *Proceedings of the 33nd conference on Winter simulation*. IEEE Computer Society, 2001, pp. 1478–1486.

[45] C. McLean and S. Leong, "**A framework for standard modular simulation**," in *Simulation Conference, 2002. Proceedings of the Winter*, vol. 2. IEEE, 2002, pp. 1613–1620.

[46] T. Meyer, C. Pöge, and G. Mayer, "**Integration of emulation functionality into an established simulation object library**," in *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference, 2012, p. 253.

[47] M. Hermann, T. Pentek, and B. Otto, "**Design principles for Industrie 4.0 scenarios: a literature review**," *Technische Universität Dortmund, Dortmund*, 2015.

[48] G. Barbieri, K. Kernschmidt, C. Fantuzzi, and B. Vogel-Heuser, "**A SysML Based Design Pattern for the High-Level Development of Mechatronic Systems to Enhance Re-Usability**," in *World Congress*, vol. 19, no. 1, 2014, pp. 3431–3437.

[49] "**Video of a cartoning machine**," Accessed date 16-February-2016. [Online]. Available: https://www.youtube.com/watch?v=_p6RyJSuHs8

[50] "**IMA Group cartoner**," Accessed date 18-February-2016. [Online]. Available: http://www.ima-industries.com/en/range-of-machines/secondary-packaging/cartoning/flexa-c/2_r20_3_14_204.html

[51] "Information on **Dunlop conveyor belting**," Accessed date 19-February-2016. [Online]. Available: http://www.dunlopconveyorbelting.com/uploads/media/Dunlop_Technical_Manual_V2.6.pdf

[52] "Information on **Prismatic Joint**," Accessed date 19-February-2016. [Online]. Available: http://it.mathworks.com/help/physmod/sm/ref/prismaticjoint.html?refresh=true