



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

*Agile Production in Software Projects for
Mission Critical Systems:
Wizard Based Product Line Lifecycle
Management carried out at National
Center for Oncological Hadrontherapy*

MASTER'S THESIS DEGREE IN
SPACE ENGINEERING - INGEGNERIA SPAZIALE

Author: **Marco Papa**

Student ID: 939767

Advisor: Prof. Consolati Giovanni

Company Thesis Manager: Dr. Casalegno Luigi

Academic Year: 2020-21

Nomenclature

AC Alternative Current

ADCS Attitude Determination & Control Systems

AIMD Active Implantable Medical Device Directive

Airbus DS Airbus Defence and Space

API Application Programming Interface

A Mass Number

B Magnetic Field Vector

CERN Conseil européen pour la recherche nucléaire

CER Clinical Evaluation Report

CE Conformité Européenne

CF2020 CNAO Framework 2020s

CNAO Centro Nazionale Adroterapia Oncologica

COMSYS Communication System

CORBA Common Object Request Broker Architecture

CPS Cyber-Physical System

CSRSD Control System Requirements Specification Document

C Carbon

DBSE Document-Based Systems Engineering

DCS Distributed Control System

DSL Domain-Specific Language

ECLSS Environmental Control and Life Support System

ECR	Electron Cyclotron Resonance
EORTC	European Organization for Research and Treatment of Cancer
EPICS	Experimental Physics and Industrial Control System
EPS	Electric Power System
EULIMA	European Light Ion Medical Accelerator
EU	European Union
E	Effectiveness Index of technical, design and operational
FEC	Front End Computer
FMEA	Failure Mode and Effects Analysis
FPGA	Field Programmable Gate Array
FWHM	Full Width at Half Maximum
GNC	Guidance Navigation Control
GRPC	Google Remote Procedure Call
GSI	Gesellschaft für Schwerionenforschung
GSPR	General Safety and Performance Requirements
G	Severity Index
HIMAC	Heavy Ion Medical Accelerator in Chiba
HIT	Heidelberg Ion-beam Therapy
HMI	Human Machine Interface
HSI	Human System Integration
HW	Hardware
H	Hydrogen
I/O	Input/Output
ICALEPCS	International Conference on Accelerator and Large Experimental Physics Control Systems
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission

IEEE Insitute of Electrical and Electronic Engineers
IH-type Interdigital H-type
INCOSE International Council on Systems Engineering
INFN Istituto Nazionale di Fisica Nucleare
IP Internet Protocol
IRR Residual Risk Index
IR Risk Index
ISO International Organization for Standardization
IT Information Technology
IVDD In Vitro Diagnostic Directive
IVDR In Vitro Diagnostic Regulation
JSON JavaScript Object Notation
LAN Local Area Network
LEBT Low Energy Beam Transport
LEP Large Electron-Positron Collider
LET Linear Energy Transfer
LINAC Linear Accelerator
MBSE Model-Based Systems Engineering
MDD Medical Device Directive
MDE Model Driven Engineering
MDR Medical Devices Regulamentation
MEBT Middle Energy Beam Transport
NASA National Aeronautics and Space Administration
NGCS Next Generation Control Systems
OAS OpenAPI Specification
OBC On Board Computer

OMA	Optimization of Medical Accelerators
OMG	Object Management Group
OPC-UA	Open Platform Communications Unified Architecture
OS	Operative System
O	Oxygen
PMCF	Post-market Clinical follow-up
PMS	Post-market surveillance
PROD	Procedure Document
PSUR	Periodic Safety Update Report
P	Probability Index
QA	Quality Assurance
QST	Quantum and Radiological Science and Technology
Q	Electron Charge
RBE	Radio-Biological Efficiency
RFQ	Radio Frequency Quadrupole
RF	Radio Frequency
RTE	Round Trip Engineering
RT	Real Time
S&M	Structures and Mechanisms
SCADA	Supervisory Control and Data Acquisition System
SOUP	Software Of Unknown Provenance
SW	Software
SySML	Systems Modeling Language
TCP	Transmission Control Protocol
TPS	Thermal Protection System
UDI	Unique Device Identification

UNI Ente Italiano di Unificazione

Abstract

Hadrontherapy is the treatment of cancer with charged carbon ion and protons beams.

Currently, the *Italian National Center for Oncology Hadrontherapy (CNAO)* aims to modernize one of the software environments of its control system: as the charged ion beams used in Hadrontherapy must be accelerated to large energies, the particle accelerators used in this treatment are complex and composed of different subsystems. Consequently, control systems are employed to supervise and control the accelerators.

A similar design would allow the integration of new types of devices into the control system as well as the introduction of new technologies into the environment. Moreover, the additive development of numerous libraries and tools may support the development of future control system applications.

The design and development of numerous software services, aimed at allowing the communication of the applications of the environment and other components of the control system, are obviously also useful in evaluating the use of the instrument in other sectors, even heterogeneous, such as aerospace.

The thesis treats the certification of medical devices. In particular, the author of the thesis carries out a critical analysis of the *Agile* development methods of software in *Mission Critical* environments, identifying the toolkit under development and describing the shorthand specification file for producing a technical file for the software. Moreover, the author creates new applications, by using a State Machine wizard, in order to simplify the steps for creating the validation documentation of a software in the CNAO Center.

Keywords: Hadrontherapy; *Agile*; JSON; CNAO; CF2020.

Sommario

L'adroterapia è il trattamento del cancro con fasci di ioni di carbonio e protoni carichi.

Attualmente, il *Centro Nazionale Italiano per l'Adroterapia Oncologica (CNAO)* ha l'obiettivo di modernizzare uno degli ambienti software del proprio sistema di controllo: poiché i fasci di ioni carichi utilizzati nell'Adroterapia devono essere accelerati a energie molto grandi, gli acceleratori di particelle utilizzati in questo trattamento sono complessi e composti da diversi sottosistemi. Di conseguenza, vengono impiegati sistemi di controllo per la supervisione e il controllo di questi acceleratori.

Un tale progetto consentirebbe l'integrazione di nuovi tipi di dispositivi nel sistema di controllo nonché l'introduzione di nuove tecnologie nell'ambiente. Inoltre, lo sviluppo additivo di numerose librerie e strumenti potrebbe supportare lo sviluppo di future applicazioni del sistema di controllo.

La progettazione e lo sviluppo di numerosi servizi software, volti a consentire la comunicazione delle applicazioni dell'ambiente e di altri componenti del sistema di controllo, sono ovviamente utili anche nella valutazione dell'utilizzo dello strumento in altri settori, anche eterogenei tra di loro, come quello aerospaziale.

L'attività di tesi si colloca nell'ambito della certificazione dei dispositivi medici. In particolare, l'autore della tesi effettua un'analisi critica delle modalità di sviluppo *Agile* di software in ambienti *Mission Critical*, identificando il toolkit in fase di sviluppo e descrivendo il file di specifiche stenografato per la produzione di un fascicolo tecnico per il software. Inoltre, l'autore della tesi ha creato nuove applicazioni, utilizzando una State Machine wizard, al fine di semplificare i passaggi per la creazione della documentazione di validazione di un software nel Centro CNAO.

Parole chiave: Adroterapia; *Agile*; JSON; CNAO; CF2020.

Contents

Nomenclature	i
Abstract	vii
Sommario	ix
Contents	xi
Introduction	1
1 Basis of the Hadrontherapy Accelerators	5
1.1 First treatments with hadron beams	5
1.2 The complexity of the particle accelerators	7
1.2.1 Accelerator designs	7
1.2.2 Accelerators for carbon ions	9
1.3 Control systems	11
1.3.1 DCS and SCADA	11
1.3.2 The <i>Standard Model</i>	12
1.3.3 Control System Hardware	13
1.3.4 Software Frameworks	14
2 CNAO - Italian National Center for Oncology Hadrontherapy	17
2.1 The first Hadrontherapy center in Italy	17
2.2 High Technology in Medical field	19
2.2.1 LEBT - Low Energy Beam Transport	19
2.2.2 RFQ and LINAC	21
2.2.3 MEBT - Middle Energy Beam Transport	21
2.2.4 The synchrotron	22
2.2.5 Extraction lines	26
2.3 Dose distribution system	28

2.4	Explanation of a cycle	29
2.5	Control System General Conceptual Model	30
2.5.1	Boundaries and layers	31
2.5.2	Component Architecture	32
2.5.3	Control System Layers Architecture	33
2.5.4	Control System Synchronisation Architecture	35
2.5.5	Control System Sequencing Architecture	36
2.5.6	Control System Communication Architecture	37
3	Medical Devices Certification	39
3.1	MDR UE 2017/745: New regulations in the medical field	39
3.1.1	Classification criteria for medical devices	40
3.2	Standard UNI IEC 62304: medical device software - life cycle processes . .	41
3.2.1	Software development process	44
3.2.2	Software maintenance process	45
3.2.3	Software risk management process	45
3.3	Standard UNI IEC ISO 14971: application of risk management to medical devices	46
3.3.1	Risk analysis	48
3.3.2	Risk evaluation	50
3.3.3	Risk control	50
3.3.4	Risk Management in CNAO projects	51
3.4	The Technical File	54
4	Integrated development environment and <i>Agile</i> development approach	57
4.1	CF2020 and the Cyber-Physical Systems	58
4.2	Product Line production process	60
4.2.1	Product Line Architecture	63
4.3	Description of the toolkit	64
4.4	Project focus	66
4.4.1	CF2020 Project Wizards	67
4.4.2	JSON Specification File	70
5	Development Software Validation Documentation	73
5.1	Wizards Application	74
5.1.1	Parallel Command Template Manager	75
5.1.2	Data Uploader Template Manager	78

5.1.3	List Monitor Template Manager	80
5.1.4	Schedule Monitor Template Manager	82
5.1.5	Device Monitor Template Manager	83
5.1.6	State Machine Template Manager	86
5.1.7	Procedure Template Manager	90
5.2	Requirements and software release	92
5.3	Results & Final Remarks	93
6	<i>Agile</i> implementation in the Aeronautics and Space field	95
6.1	Systems engineering	95
6.2	The importance of the V-Model	97
6.3	Model-Based Systems Engineering	97
6.3.1	Main features and motivation	98
6.3.2	Benefits implementation	99
6.3.3	MBSE in the present projects	100
6.4	Final remarks	101
7	Conclusions and future developments	103
	Bibliography	105
A	Gantt Chart	111
B	System Architecture	113
C	Overview of the risk management process for medical devices	115
D	JSON Schema snippets	117
	List of Figures	129
	List of Tables	131
	Acknowledgements	133

Introduction

The operation of particle therapy facilities requires complex control and supervision systems, often spanning several software development environments, each containing many applications. While the component-based development approach brings many benefits, the integration of the parts is still left to the initiative of each developer without a repeatable path that can be demonstrated when the artifacts must undergo the certification process.

Background information

Particle therapy is a cancer treatment method that is becoming increasingly more prominent. In contrast to Radiotherapy, in which radiation is used to damage cancer cells, in particle therapy ions are accelerated to deposit energy in the cells. The main advantage of Hadrontherapy over Radiotherapy is the former's dose deposition profile, which allows a lower dose deposition in healthy tissue surrounding the targeted tumor. However, Hadrontherapy requires the use of a particle accelerator of large dimensions and complexity. The large quantity of equipment involved in generating the particle beam and applying it in patient treatment requires an elaboration control system for the facility.

The Optimization of Medical Accelerators (OMA) project is a European Training Network funded by the European Union's Horizon 2020 research and innovation programme. This project, started in 2016, joins several institutions, such as universities, research centers, and Hadrontherapy facilities, to perform projects under a common set of goals. The network created by the project, which contains more than 30 organizations, is composed of beneficiaries, partner organizations, and adjunct partners. The goals of the network are separated into a total of three work packages. When writing the proposal, each beneficiary organization proposed a project, with the scope aligned with one of the work packages, and later received funds to hire a research fellow to work on the project for a duration of three years.

The CNAO was in the position of hosting two of the OMA network's projects: the project named *Tumor Tracking in particle therapy* with the supervision of Prof. Guido Baroni and

a project entitled *Light ion therapy software for data exchange* that is the one underlying the work performed in the context thesis, with the supervision of Prof. Luigi Casalegno. The project is based on creating a common software bus that enables present and future packages of the configuration and support environment of the control system to easily interconnect in a complex and widely distributed Hadrontherapy facility.[1]

An integral part of the project is the specification of protocols to be used in the facility's control system configuration and support environment. These protocols should allow data exchange, assurance of security and privacy, and discovery of devices. In order to support the development of new applications of the configuration and support environment, a Product Line Architecture was designed. The architecture specifies the scope of applications, general layout, and variation points. Additionally, a wizard generator has been developed to create customized base applications in accordance to the product-line architecture.

One of the major concerns in the medical environment is the assurance that programs and components are functioning as per specification. In fact, novel control systems for a medical accelerator facility demand an increased monitoring via specifying tools that help with visualization and construction of workflows.

Moreover, the thesis has concerns about certifying the medical software, aiming at improving the certification of applications created for this project.

Analysis of the work

The main objective of the author is planning and performing the upgrade of a control system environment of a Hadrontherapy facility. The software in this environment performs a support role in the medical facility, and is used for tasks such as configuring the accelerator systems, and supporting the clinical workflow. As part of the upgrade, the author chooses new technologies for the environment. Moreover, a Product Line Architecture is designed by the author to define the general architecture of future applications. In addition, new applications which the CNAO center will use are created by the author, using a State Machine wizard.

Therefore, one of the important part of the work is based on studying the technical regulations in order to lead the programmer towards the development of a software that meets the performance and safety requirements, in accordance with the provisions of the law. Associating the normative part with the technical one, the essential elements of a validation documentation should be elaborated.

Structure and Contents

The thesis is organised in seven chapters:

1. Chapter 1 describes the physical basis of the particle accelerators, highlighting the functionality of the accelerator control system.
2. Chapter 2 focuses on the description of the CNAO center and on the detailed description of the operating principle of the synchrotron facility within the structure.
3. Chapter 3 presents a description of the technical and safety regulations for the realization of the technical files in order to lead the programmer towards the development of a software that complies with the performance and safety requirements, in accordance with the provisions of the law.
4. In chapter 4, the author carries out a new product line approach in detail, as well as the designed architecture for the applications of the environment that envisages the presence of standard services that are to be used by the applications. The design of the Product Line Architecture was heavily influenced by the mandatory medical software certification processes that applications in the upgraded environment must undergo. Moreover, the author details the design tactics adopted in the Product Line Architecture to aid the certification process.
5. In chapter 5, the author presents a new State Machine wizard, with new applications of the upgraded environment made by the author. They are made in order to improve a system already existing in the CNAO Center, using an iterative approach to verify that all documentation is coherent with the work.
6. Chapter 6 describes a project modeling methodology used in the aerospace field, pointing out the importance of the *Agile* philosophy in this area. Lastly, the author underlines the similarities of the approaches used in the CNAO Center toolkit and in the application of modeling in the aerospace engineering.
7. Chapter 7 evaluates the work, focusing both on the impact of individual components, as well as on the contribution of the upgraded environment to the control system as a whole concludes the thesis, summarizing the work performed and the achieved results. Afterwards, a brief description of the expected continuation of this work is made, including the areas where this work can be expanded upon.

All the activities carried out in the CNAO Center by the author of the thesis are described in a Gantt Chart, reported in the Appendix A.

1 | Basis of the Hadrontherapy Accelerators

In the last 60 years, Hadrontherapy has made great advances passing from a stage of pure research to a well-established treatment modality for solid tumours.

Hadrontherapy is a specific type of oncological Radiotherapy, which makes use of fast hadrons (non-elementary particles made of quarks and antiquarks). Today only two types of hadrons are used to treat solid tumours, protons and fully stripped carbon ions, but other hadrons, such as neutrons, charged pions, antiprotons, helium ions (i.e. alpha particles) and other light ions nuclei (as lithium, oxygen and silicon ions) have been either used or planned to be used for the treatment of tumour patients.

Protons are nowadays an important tool in clinical practice due to approximately fifty hospital-based centres in operation and to the continuous increasing number of facilities proposed worldwide. Moreover, very promising results have been obtained with carbon ion beams, especially in the treatment of specific radio-resistant tumours. However, clinical trials are still needed to define the tumours types to be treated and to optimize the protocols.[2]

This therapy based on hadrons is a multi-disciplinary field in which physics, biology, medicine and engineering meet. The continuous technological improvements of medical particle accelerators are the result of the development of research accelerators used in nuclear and high-energy physics.

1.1. First treatments with hadron beams

Robert Bob Wilson is the father of Hadrontherapy for his 1946 seminal paper[2] in which, after realizing that depth-dose profiles of protons in matter have a significant increase at the end of their range (the so-called “Bragg Peak”, as shown in Fig.1.1), he proposed to make use of this property for treating deep seated tumours. Nevertheless, this important observation would not have developed into a very effective and not invasive way of treating

some type of solid tumours if three important discoveries and inventions had not been made in the fifty years preceding Wilson’s work.

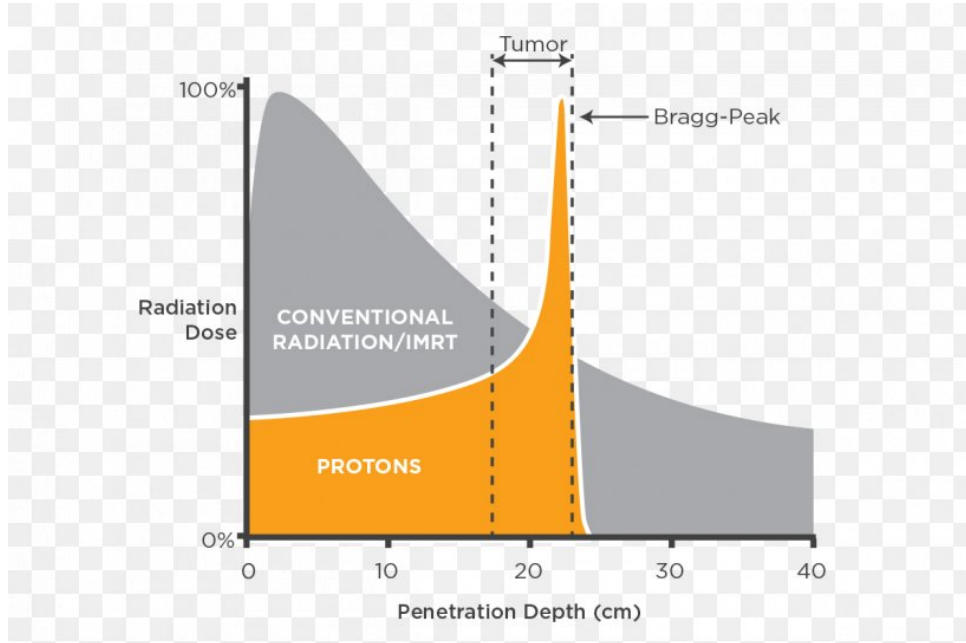


Figure 1.1: Dose-depth curves of different beams[3]

The event that opened the way to Hadrontherapy was in 1929 the invention of the cyclotron by Ernest Orlando Lawrence. Lawrence had the idea of adding a magnetic field to the linear accelerator, previously proposed by Wideröe.[2]

The first hadrons used in Radiotherapy, soon after the construction of the first cyclotron by Lawrence and Livingstone, were fast neutrons. In 1936, the first experimental studies were published by the Lawrence brothers and, at the end of September 1938, the first patients were treated with neutrons. These were produced with the more powerful 37-inch cyclotron which accelerated deuterons up to 8 MeV, by bombarding a beryllium target.[2] In the traversed tissues the neutrons transfer their energy mainly to highly ionizing protons, which have a value of the “Linear Energy Transfer” (LET) that is much larger than the one of the electrons put in motion by MeV photons produced by LINACs. This is a favourable feature when treating radio-resistant tumours¹.

However, neutron beams have an unfavourable depth-dose distribution and are difficult to collimate; these are the reasons for which they are no longer used in the treatment of radio-resistant tumours and have been substituted, in 1954, with proton beams. Treatments

¹Radio-resistant tumours are less sensitive to X-rays than the surrounding normal tissues.

with other charged ions soon followed, with the usage of helium in 1957, neon ions in 1975, and carbon ions in 1994.[4][2]

1.2. The complexity of the particle accelerators

Large particle accelerators, whether medical or research oriented, are complex systems that can only be developed by combining the efforts of personnel from several heterogeneous fields. When designing a complex accelerator, the earliest obtainable architecturally significant requirements are originated by the users of the system, such as researchers performing scientific experiments, or healthcare providers in the treatment of patients.

These initial system requirements have to be analysed by personnel knowledgeable in particle physics and the domain of accelerator design. Additionally, other important stakeholders also need to be consulted, such as accelerator operators and maintenance staff.

1.2.1. Accelerator designs

Various accelerator designs have been used in Hadrontherapy. The most common designs are linear accelerators, cyclotrons, and synchrotrons. Linear electron accelerators are widely used in X-ray Radiotherapy. While proton linear accelerator design has first been proposed in 1989 by Lennox et al.[5], currently there are no Hadrontherapy facilities using linear accelerators.[4] There are currently several proposed designs for particle therapy linear accelerators, such as the ADAM company proton linear accelerator design,[6] and the TERA foundation carbon cyclotron-to-linear-accelerator design.[7] These accelerators allow beams with high repetition rates, with over 100 pulses per second, enabling fast energy modulation, as the energy can be adjusted every pulse. Drawbacks of these designs include the accelerator length, and costs.[8]

The most common accelerator design currently used in Hadrontherapy facilities is the cyclotron, with several companies offering commercial solutions, such as IBA,[9] and Varian.[10][4] In cyclotrons, charged particles are subjected to a constant magnetic field, and accelerated in an acceleration gap controlled via a RF frequency. As the bunched particles are accelerated, the radius of their trajectory increases. Once the radius of the trajectory of the bunch matches the cyclotron extraction window, the bunch passes through the window and is extracted. The resulting extracted bunches from cyclotrons have a periodicity so fast to be considered a constant beam for the purposes of cancer treatment.

Because of their design, cyclotrons produce beams with constant kinetic energy, thus requiring a separate mechanism for energy modulation.²[8] The energy modulation is usually performed with an external energy selection system that reduces the energy of the beam. The extra energy modulation step affects the beam quality in multiple ways. Firstly, the beam shape is modified, and debris is created as result of the collision with the absorber material. Secondly, the energy modulation system process takes a slight amount of time to physically move the absorber wedges, leading to a short delay. In cases of high beam energy attenuation for protons, or attenuation of carbon ion beams, the nuclear interactions may cause the energy selection system area to become radioactive.[8]

Cyclotrons designs have two common variants: the isochronous variant and the synchrocyclotron variant. These variants behave similarly but, unlike the isochronous version where the RF frequency is constant, in synchrocyclotrons the RF frequency decreases during the acceleration.[12]

Another common accelerator design for particle therapy is the synchrotron. In the synchrotron design, the particle beam repeatedly travels a circular path as it is being accelerated. Along the circular path, several magnets perform the bending of the beam and beam compression. Once the particle bunch reaches the desired energy, an extraction mechanism is activated and the beam is delivered.[12]

Synchrotron designs usually provide low beam periodicity rate, with conventional designs taking more than one second to reduce the magnetic field, prepare a new beam, and accelerate it to the desired energy. As noted by Amaldi et al.[8], the periodicity of synchrotrons may present issues in the treatment of moving tumors, as it may roughly coincide with patients breathing patterns. Recently, novel designs are being proposed, offering much faster repetition rates for proton synchrotrons.

Overall, cyclotron and synchrotron designs are the current norm in the field of particle therapy. Cyclotron designs offer benefits such as lower acquisition cost, simpler and less costly operation, smaller size, and provide a continuous beam. However, there are currently no fully functioning cyclotrons designs capable of accelerating carbon ions at the energy necessary for Hadrontherapy.[4] On the other hand, synchrotron designs are more versatile, allowing the accelerated energy to be determined, every acceleration cycle, and thus not requiring absorbers to modulate energy.

Currently, there are several accelerator designs aimed at addressing issues and improving

²In various embodiments, a radiation therapy system can include a cyclotron that outputs a charged particle beam. In addition, the radiation therapy system can include an apparatus to receive the charged particle beam from the cyclotron. This apparatus decelerates or further accelerates the charged particle beam to produce a reduced or increased energy charged particle beam.[11]

Hadrontherapy, in various development stages. Super cooling technologies have already been proposed and successfully implemented in order to reduce the size and weight of cyclotrons and synchrotrons.[4]

1.2.2. Accelerators for carbon ions

A beam of carbon ions is by its nature a very different radiation compared to X-rays and protons. Firstly, a fully stripped carbon ion, a nucleus of carbon made of six protons and six neutrons, produces a sharper Bragg Peak than the one produced by protons, so that the spot due to a mono-energetic carbon ion beam has a FWHM³ of 3-4 mm instead of 10 mm.[2] This implies that with carbon ions one can deliver a dose that is macroscopically more conformal to the tumour target than the doses due to protons and X-rays.

Secondly, carbon ions, with an electric charge six times larger than protons, slow down more rapidly.⁴ In this process they leave more energy per unit length than a proton in the ratio of the energy needed to reach the same depth (about 23).[2] This is the average ratio of the LET of the two beams and is also the ratio between the numbers of ionizations left in the nucleus of each traversed cell. The consequence is that, even if a beam of carbon ions deposits the same dose in a tumour tissue, at the molecular level the biological effects are different from those of protons (and of X-rays) because they have a larger Radio-Biological Efficiency (RBE).

Carbon ions at the end of their range can have, in some types of cells, three times higher RBE than protons. This makes them more effective in killing radio-resistant cells than protons and X-rays. This is similar to what happens with fast neutron beams but with the advantage that the macroscopic distribution of the dose can be made very conformal to the tumour target, because of the small dimensions of the spot. Thus, the high LET difference makes carbon ions capable of treating radio-resistant and hypoxic tumours which represent approximately 5% of the 2000 tumours irradiated every year with X-rays, in a population of one million inhabitants.

In 1994, in Japan, Yasuo Hirao and collaborators proposed the realization of the Heavy Ion Medical Accelerator in Chiba (HIMAC) to be built in the Chiba prefecture. In 1994 the facility treated the first patient with carbon ions at a maximum energy of 4800 MeV. By the end of 2014, under the leadership of Hirohito Tsujii, about 9000 patients had been treated and many difficult and common tumours had been shown to be controllable.[2]

³In a distribution, full width at half maximum (FWHM) is the difference between the two values of the independent variable at which the dependent variable is equal to half of its maximum value.

⁴For example, to reach a 28 cm depth, a carbon ion must have initially about 4800 MeV (400 MeV/nucleon) instead of the 210 MeV of a proton.[2]

In 1993, Gerhard Kraft obtained the approval for the construction of a carbon ion facility at GSI (Darmstadt). Treatments started in 1997 and about 400 patients were treated with carbon ion beams under the direction of the radiation oncologist Jürgen Debus.[2]

In 2009, the brand new Heidelberg Ion-beam Therapy (HIT) Centre was opened, where all the medical and technical competences accumulated in Darmstadt were applied. The HIT was a joint endeavour of the GSI and the Siemens Medical company. HIT, with its 25 m long rotating gantry to be used both for protons and carbon ions, has been the first Hadrontherapy centre able to compare clinical results obtained with protons and carbon ions with beams coming from the optimal directions. After HIT, CNAO (Centro Nazionale di Adroterapia Oncologica) in Pavia was the second institution in the world to offer, outside Japan, Radiotherapy treatment with carbon ions.[2]

Japan remains at the forefront of ion therapy with approximately 10,000 patients, while in 2013 the National Cancer Institute and the Department of Energy in the USA decided to launch a study for an American carbon ion and proton centre.[2][4]

Country	Center	Location	Max. Energy per Nucleon (MeV)
Austria	MedAustron	Wiener Neustadt	403/u
China	HICTC	Wuwei, Gansu	400/u
China	SPHIC	Shanghai	430/u
Germany	HIT	Heidelberg	430/u
Germany	MIT	Marburg	430/u
Italy	CNAO	Pavia	480/u
Japan	HIMAC, QST	Chiba	800/u
Japan	HIBMC	Hyogo	320/u
Japan	SAGA-HIMAT	Tosu	400/u
Japan	GHMC	Gunma	400/u
Japan	i-RKCC	Yokohama	430/u
Japan	OSHITC	Osaka	430/u

Table 1.1: Carbon particle therapy facilities in clinical operation.[12]

Tab.1.1 presents a list of all carbon therapy facilities currently in operation, as well as their respective location and maximum energy per nucleon. Today, all carbon ion particle therapy facilities use synchrotron accelerator designs.[12][4]

1.3. Control systems

In the field of particle accelerator design, the term control system is used more broadly than in some other industrial engineering domains.

The role of an accelerator control system is to supervise all devices and subsystems of the accelerator, taking into account the states and transitions of the system. At a software level, an accelerator control system should be able to perform error detection, and follow up by providing proper recovery methods. Additionally, a mapping must be made from all operation requirements of the accelerator into control system operation modes, allowing operators to fulfill the requirements by selecting the appropriate operation modes. As a result of these requirements and the complexity of medical and experimental accelerators, these control systems are composed of hardware, networking components, off-the-shelf, and tailor made software.[13]

The ICALEPCS⁵ biennial conference, which gathers control system specialists from around the world, interprets the scope of the term “control systems” to include the following:

- “All components or functions, such as processors, interfaces, fieldbusses, networks, human interfaces, system and application software, algorithms, architectures, databases, etc...”
- "All aspects of these components, including engineering, execution methodologies, project management, costs, etc..."[14]

1.3.1. DCS and SCADA

A commonly presented question is the difference between a Distributed Control System (DCS) and a Supervisory Control and Data Acquisition system (SCADA). These terms are used to categorize control systems, or parts of control systems. The difficulty of differentiating these terms is often higher in the domain of accelerator control because of the frequent usage of custom software and hardware in control systems in order to achieve the domain’s particular requirements. In the literature, SCADA systems often do not refer to full control systems, but to a layer exclusively made of software packages, that rests above a lower layer of a control system. Under this interpretation, the main purpose of the SCADA system is to acquire data from the lower layer components and provide supervision capabilities.[4]

⁵The International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS) is a biennial series of conferences inaugurated by a number of control system specialists from accelerator laboratories around the world.

DCS is similar to SCADA as both communicate with lower layers, and provide a centralized interface for operators. However, DCSs are more closely connected to the hardware, using process driven communication rather than event driven, and are often being sold together as a packaged solution.[4]

1.3.2. The *Standard Model*

The control system *Standard Model* was proposed by Kuiper in 1991.[15] This model proposes a separation of concerns into control system tiers. For each tier, hardware components are specified, as well as communication between components, and allocation of software applications, as shown in Fig.1.2.[16]

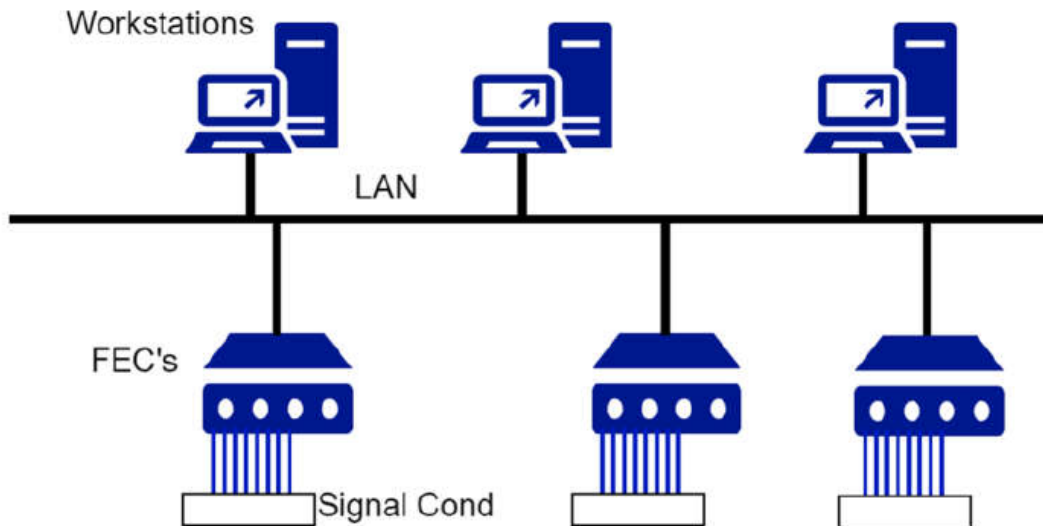


Figure 1.2: Kuiper's *Standard Model*[15]

The *Standard Model* separates accelerator control systems in three tiers.

1. The first tier consists of workstation machines that provide the user interface to operators. The workstations display the current status of the control system, current and historical data measurements, alarms, as well as any other domain specific information operators require.
2. The second tier transports data between the first and third tier of the control system using a local area network (LAN).
3. The third and the last tier is responsible for implementing the domain logic of

the control system. Devices in this tier⁶ are responsible for the acquisition and processing of distributed data, supervision of the control system, as well as sequential and closed-loop control. This tier is the only one that communicates directly with the accelerator hardware.

The standard model also recognizes the importance of ensuring the correlation of data, as accelerators are distributed systems with strong real time constraints. Three approaches for correlating measurement data presented are:[4]

- Package data along with a correlation identifier generated at the source of the data, such as a time stamp.
- Synchronize the data collection system, in such a way that all data collection is triggered simultaneously.
- Determine a single component that collects all data.

Several variations of the standard model have been proposed to address challenges encountered in accelerator control systems. These solutions usually address specific requirements, and extend the standard model.

To address these requirements, an extension has been proposed to the standard model, adding multiple network tiers and using different infrastructure to address challenges in each tier. For subsystems with stronger real time constraints, a reflected memory scheme is added to several controllers to allow these controllers to have necessary data for operation in a short time window.[4]

1.3.3. Control System Hardware

Important requirements for an accelerator control system are dependability, fault tolerance, predictability, extensibility, and usability. In the Hadrontherapy domain, additional importance should be given to fault tolerance, dependability, and safety, as system failures pose unacceptable risk to patients and medical personnel. On facilities developed by commercial Hadrontherapy system providers, that provide a catalogue of turn-key Hadrontherapy solutions, the importance of extensibility is often lowered. In the medical domain, the usability requirements should be reinforced as poorly designed interfaces may lead to incorrect usage by trained operators, resulting in errors which may cause permanent and irreparable harm.[4]

⁶In Kuiper's article the third layer devices are referred as Front End (Micro)Computers, or FEC's. Meanwhile, computational devices have changed considerably and the term "front-end" is more commonly used to refer to devices that interface with the end user.[15]

Upper tier In the accelerator control system domain, upper tier of control system present the least amount of non-standard hardware requirements. Müller states that control equipment at this tier only differs from regular IT equipment by requiring assurances regarding availability. Modern control system software at this tier often is able to run on all major operating systems.[13]

Lower tiers The lower tiers of the control system are often the most varied. In systems following the standard model, the lower tier should contain "Front-End Computers" or microprocessors. This tier is the closest to the accelerator, and so usually possesses the strictest real time constraints. Devices in this tier are also responsible for obtaining sensor data, and if possible perform processing over them before sending the data to the upper tiers. Additionally, in many accelerator facilities the data correlation process occurs in this tier, by appending sensor measurements with a time stamp or cycle number for synchronization.

Networking In the network tier, control system equipment differs sharply from conventional network equipment. The most visible difference is often the network hierarchy, with control networks often having much deeper architecture, involving additional layers. Moreover, these different networks layers with different protocols allow communication between instruments to controllers, and then to operator interface equipment.

The network traffic patterns in control systems are also inevitably dissimilar from conventional networks. Control system network traffic must account for periodic data in the form of sensor measurement data, alarms, and state messages. Besides, real-time constraints in the order of microseconds must be accounted for designing the network. In order to fulfil real time requirements, custom network and I/O solutions are often used.[4]

1.3.4. Software Frameworks

Particle accelerator control systems often follow patterns that can be leveraged to reuse concepts and architecture designs. Unsurprisingly, the same also occurs for control system software.

In the past, several accelerator teams proposed control system software designs, while seeking collaboration partners for the development of reusable toolkits for the construction of accelerator control systems. From these joint efforts, several successful open source frameworks for control systems were developed, such as EPICS and TANGO. Additionally, several commercial options have been made available for usage in accelerator facilities.

Experimental Physics and Industrial Control System (EPICS) The EPICS architecture is the oldest control system framework designed for the particle accelerator domain.[17]

It is a distributed control system framework that runs on top of an infrastructure composed of workstations for operators, I/O controller computers, and file servers, connected via a LAN connection. At the time, the framework provided a distributed database, as well as several subsystems necessary for accelerator control, such as alarm management, archiving, sequencing, and operator display. The EPICS software framework was designed based on the following requirements:[17][4]

- Providing standard functionality required in an accelerator control system.
- Reducing time required for control system development and alterations.
- Allowing extensibility of the framework.

TANGO Framework TANGO is an object oriented, distributed control system framework which defines a communication protocol, an Application Programmers Interface (API) and provides a set of tools and libraries to build software for control systems, especially SCADA. The TANGO framework is multi-platform and supports the Java, C++, and Python programming languages.[18]

TANGO was designed to manage small and large systems. Each system has a centralised database that stores configuration data used at startup of a device server, and acts as name server by storing the dynamic network addresses. The database acts as a permanent store of dynamic settings which need to be memorised. Each TANGO system has a database and is identified by its TANGO host. A large system can be set up of more than ten thousand devices (the limit has not been reached yet) supported by a communication protocol that defines how all components of the system communicate with each other. Tango uses CORBA⁷ for synchronous communications and ZeroMQ for asynchronous communications. The detail of these protocols are hidden from the developer and user of TANGO by the API and high level tools.

⁷The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) designed to facilitate the communication of systems that are deployed on diverse platforms. CORBA enables collaboration between systems on different operating systems, programming languages, and computing hardware.[19]

2 | CNAO - Italian National Center for Oncology Hadrontherapy

The National Center for Oncology Hadrontherapy (CNAO), in Pavia, is a facility dedicated to Hadrontherapy. The advantages of Radiotherapy with hadrons, compared to conventional Radiotherapy with photons, are of two types: geometric and radiobiological.

Geometrically, Hadrontherapy allows dose delivery to the tumor while reducing the dose delivered to surrounding healthy tissue. From the radiobiological point of view, protons do not offer a real advantage over photons, while carbon ions deposit more energy per unit length and are, therefore, able to produce damage more hardly to fix to tumor tissues. This results in a relative biological effect about three times bigger than photons at the end of the pathway, where the tumor to be eliminated is located.

2.1. The first Hadrontherapy center in Italy

The history of the National Center of Oncological Hadrontherapy (CNAO) began when a report called *For a Centre of Teletherapy with Hadrons* signed by Giampiero Tosi and Ugo Amaldi was published in May 1991.[20]

Tosi was a well-known Italian medical physicist and the director of the Medical Physics Department of the Niguarda Hospital (Milan, Italy). Ugo Amaldi, a particle accelerator physicist, was a former member of the *Istituto Superiore di Sanità* and, at that time, he was working at the Geneva-based CERN where he headed a collaboration project of about five hundred physicists to create and use one of the four main experiments with the LEP accelerator.

The 1991 the report called the attention of Nicola Cabibbo,[20] who was the President of INFN at that time. Therefore, in 1992 initial funding was obtained to study a new accelerator that could accelerate both protons and light ions to be used in the treatment

of deep tumours. The study, which was funded by INFN, was called ATER, short for the Italian "AdroTERapia", the word coined by Amaldi for this new type of advanced radiation therapy.

Amaldi states that the EULIMA project,¹ financed by the European Commission, motivated several national Hadrontherapy projects, such as the one in Italy.[8]

The TERA foundation was founded in 1992, and was the main contributor to what would become the CNAO Foundation project. Headquartered in Novara, Italy, the TERA foundation would come to staff over 170 employees in the Hadrontherapy domain, such as physicists, engineers, and technicians.[20] After 10 years, the TERA foundation authored three Hadrontherapy facility design proposals. The first two proposed facilities aimed for the city of Novara, in Italy, while the later aimed at Milan. While the initial proposals were eventually unsuccessful in securing funding for building the proposed Hadrontherapy facility, a study by CERN (with collaboration of TERA, MedAustron, Oncology 2000, and GSI[22]), was performed in the year 2000, and would provide the TERA the groundwork for the proposal which would later become the CNAO foundation.

Regarding technological decisions, the EULIMA project recommended the use of a synchrotron. At that point, oxygen was proposed as the heavy ion of choice, but later scientific consensus indicated that carbon ions instead would be a better choice.

The construction phase, started in 2001, saw the concrete designs for the facility finalized, and commissioned. The CNAO foundation received, as a result of a signed agreement between the two organizations, extensive documentation from the TERA foundation as well as intellectual property. The plot of land where the CNAO facility was built, next to the San Matteo general hospital of Pavia, was granted free of charge by the Province of Pavia.[20]

During this phase, in 2003,[20] the design of the control system began. In order to do this, the accelerator specification was analyzed, resulting in a set of requirement documents for the control system. From these requirements, along with documentation detailing a set of best practices and standards, the control system, and its sub-systems were designed and developed, obtaining, in 2013, the CE certification label.

¹At the European Organization for Research and Treatment of Cancer (EORTC), meeting held in Nice, on October 18-19, 1985, the idea of a European high energy Light Ion Medical Accelerator (EULIMA) had been proposed. On March 14, 1986 a group of potential medical users, radiobiologists, physicists and accelerator engineers from different European countries had met at CERN to define the goals of the project and to examine the possibility to carry out a feasibility study. A request for funding for such a study had been introduced to the European Economic Community.[21]

2.2. High Technology in Medical field

Accelerator machines much larger and more complex than a commercial LINAC for Radiotherapy with X-rays² are used to accelerate protons and carbon ions.

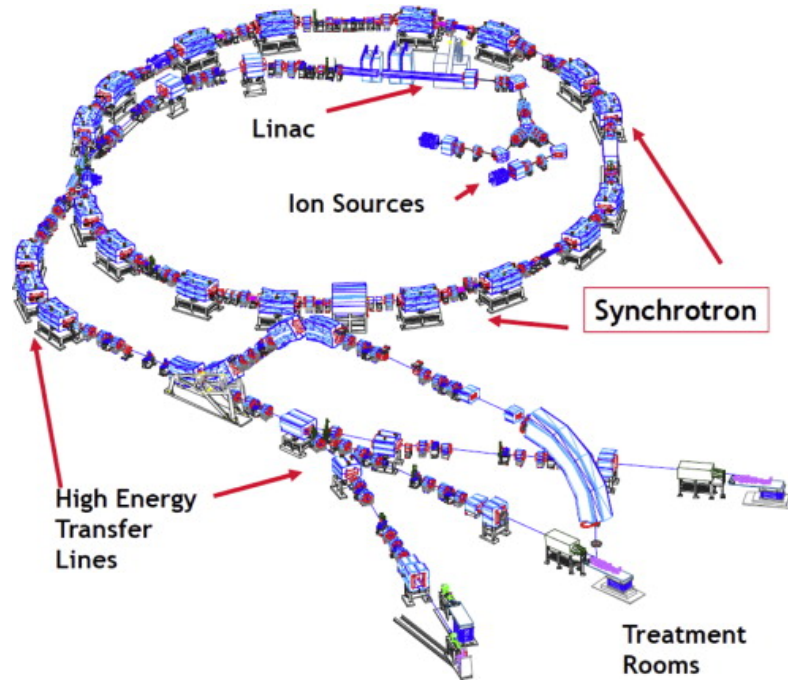


Figure 2.1: CNAO facilities[23]

The main accelerator of the CNAO is a synchrotron, a circular accelerator, about 25 meters in diameter, inside which the injection lines and the linear pre-accelerator are housed.[24] Outside the main ring there are four extraction lines, about 50 meters each, which bring the extracted beam in three treatment rooms. Therefore, a horizontal and a vertical line arrive in the central room. A clear image of the main CNAO facilities is shown in Fig.2.1.

2.2.1. LEBT - Low Energy Beam Transport

The LEBT is the line that carries the energy beam from the sources to the Radio-Frequency Quadrupole (RFQ).

There are actually two different LEBT settings, one for protons and one for carbon ions. For each source there is a dedicated initial section which has the purpose of selecting the

²To have an idea of the difference in energy required to particles, the energy of electrons used in Radiotherapy is of the order of 5÷15 MeV, while the energy required to carbon ions for deep treatments at CNAO is 4800 MeV.

desired species and allowing the observation of the beam and the adjustment of the source itself while using the other source.[24]

A second section is common to the two lines and includes a chopper which has the purpose of cutting the portion of the energy beam useful for acceleration in the LINAC (Linear Accelerator) and injection in the synchrotron. In this way the aim of minimizing the beam losses in the RFQ is also achieved.

The two sources are of the ECR (Electron Cyclotron Resonance) type and are identical to each other.[24] Each source is capable of producing multiple types of particle beams by simply changing the gas used to produce the plasma and optimizing source settings, such as the power RF and the potentials of the extraction electrodes. In this way, each source is able to make up for the absence of the other. In normal operation, each source produces only one type of energy beam which allows the accelerated ion type to be changed in a short time. The geometry of the sources and LEBT is illustrated in Fig.2.2.

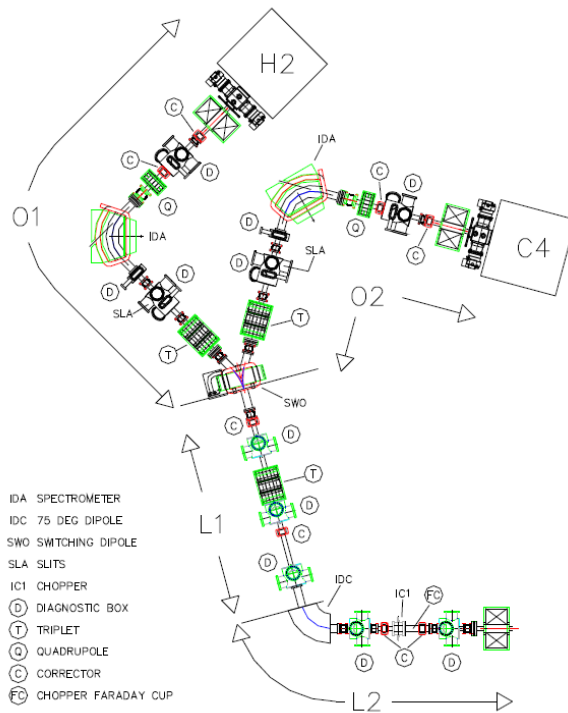


Figure 2.2: LEBT layout[24]

To allow easier selection of the desired species, the ions $^{12}\text{C}^{4+}$ e H_3^+ are used, because they have the same charge-to-mass-number ratio ($Q/A = 1/3$). The oxygen ion with the closest magnetic rigidity is the $^{16}\text{O}^{5+}$, whose magnetic rigidity³ differs with approximately 3% from that of the $^{12}\text{C}^{4+}$. Therefore, the minimum resolution required for effective selection of the desired species is about 30. The beam energy required at the RFQ input, and consequently the beam energy in the LEBT, is 8 keV/u.[24]

³The magnetic rigidity is a measure of the momentum of an electric particle moving normally across a magnetic field (as in a cyclotron) equal to the product of the radius of curvature by the intensity of the field.

2.2.2. RFQ and LINAC

The LINAC and the RFQ of CNAO are realized in collaboration with the GSI in Darmstadt and are similar to those installed in the center of Heidelberg. The RFQ must provide the appropriate energy beam to be accelerated by the LINAC. It must therefore accelerate, longitudinally pack and transversely adapt the beam. The energy at the input of the RFQ, as already mentioned, is 8 keV/u while the energy at the output must be 400 keV/u. Moreover, there is a matching section that contains a pair of correctors, a quadrupole doublet and a measuring station in only 200 mm.[24]

After the RFQ, there is the LINAC. This is an IH-type structure that accelerates the beam from 400 keV/u to 7 MeV/u in a single 3.8 m long tank.[24] The structure was designed to achieve a particular combination of acceleration and focusing, known as KONUS. The voltage at the gap along the accelerator increases with the length of the gap, maintaining a constant gradient of about 18 MV/m.[24]

The LINAC consists of four KONUS sections between which are housed three triplets of quadrupoles. Immediately after the LINAC tank, there is a fourth triplet that serves to focus the energy beam on the stripping sheet. The latter has the purpose of tearing the remaining electrons from the particles and thus producing particle beams of C⁶⁺ and protons.

The duration of the energy beam emitted by the LINAC is maximum 200 μ s.[24] The synchrotron, considering the time of acceleration, extraction and the time needed to return to initial conditions, needs a energy beam every 2-3 s. To avoid the variation of temperature (and consequently of resonant frequency) that would follow a non-periodic use of the LINAC, it is necessary to pulse the LINAC even when it is not needed to produce a beam. For this, a power into the cavity at a repetition rate of 5 Hz is emitted.[24] The chopper of the LEBT will take care of not letting the beam through when it is not needed.

2.2.3. MEBT - Middle Energy Beam Transport

The MEBT, shown in Fig.2.3, is the line that transports the beam from the stripping sheet to the injection point in the synchrotron. In addition to the task of transporting the energy beam while minimizing losses and harmonizing the shape of the beam to the desired shape, the MEBT must guarantee some other tasks:

- misalign the beam vertically to achieve the desired emittance dilution;
- adjust the injection angle to optimize the multi-turn injection process;

- conclude the process of species selection by eliminating any ions that, before stripping, had $A/Q = 3$ and after stripping do not have $A/Q = 2$;
- change the injected current;
- optimize the momentum dispersion of the injected beam.

In order to ensure these tasks, slits in the dispersive zone, a debuncher, and a set of variable transparency gratings were inserted into the MEBT.

The injected current is varied by inserting a grid into the beam path whose transparency can be chosen from 50%, 20% and 10%.[24] The use of a grid instead of a collimator has the advantage of leaving the global dimensions of the beam unchanged regardless of the transmitted current, thus simplifying the optimization of the injection process.

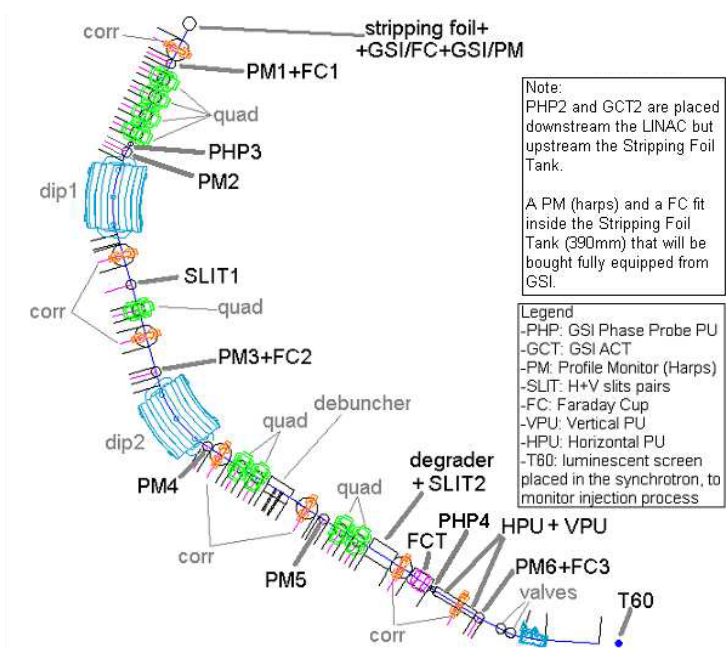


Figure 2.3: MEBT layout[24]

2.2.4. The synchrotron

The main accelerator that is used in CNAO is a synchrotron. The beam is injected at 7 MeV/u and is accelerated to energies between 60 MeV (minimum energy for the protons) and 400 MeV/u (maximum energy for the carbon beams). The magnetic lattice, that is the sequence of magnets that make up the accelerator, is based on two symmetrical and achromatic arcs that have been detuned and joined by two straight non-dispersive sections.[24]

There are 16 dipoles and 24 quadrupoles in the synchrotron. The 16 dipoles are powered in series by a single power supply, while the quadrupoles have been divided into three families of 8 quadrupoles each. The three families of quadrupoles allow the flexibility necessary to obtain all the required tune values while keeping the two regions non-dispersive.[24]

Injection The injection scheme chosen for the CNAO is of the multi-turn injection type in the horizontal plane. In order to meet the requirements of space charge and

beta function values to the patient in the extracted beam, it was decided to increase the vertical emittance of the beam in the synchrotron. This is achieved with a vertical misalignment between the closed orbit of the synchrotron and the entry trajectory of the injected beam. The filamentation resulting from this misalignment causes the desired increase in emittance.

The multi-turn injection occurs by creating a local distortion of the closed orbit, commonly called bump, that brings this orbit closer to the electrostatic injection septum. The amplitude of the bump is then gradually decreased to zero. This way, the injected beam rolls up spiraling around the closed orbit.

Since the therapy requires maximum stability and repeatability of the beam, the particles, that after injection are outside the design emittance, will be eliminated using two scrapers. In order to minimize mechanical movement, beam scraping is done by moving the beam with a bump, rather than moving the scrapers.

Magnetic cycle In a synchrotron, the beam trajectory is constant. This implies that as the beam is accelerated, the magnetic field must increase to provide the same deflection to increasing energetic particles. The choice for the magnetic cycle is the result of a compromise between the duration of the treatment and the required power. The trend of a magnetic cycle of the synchrotron dipoles is illustrated in Fig.2.4.

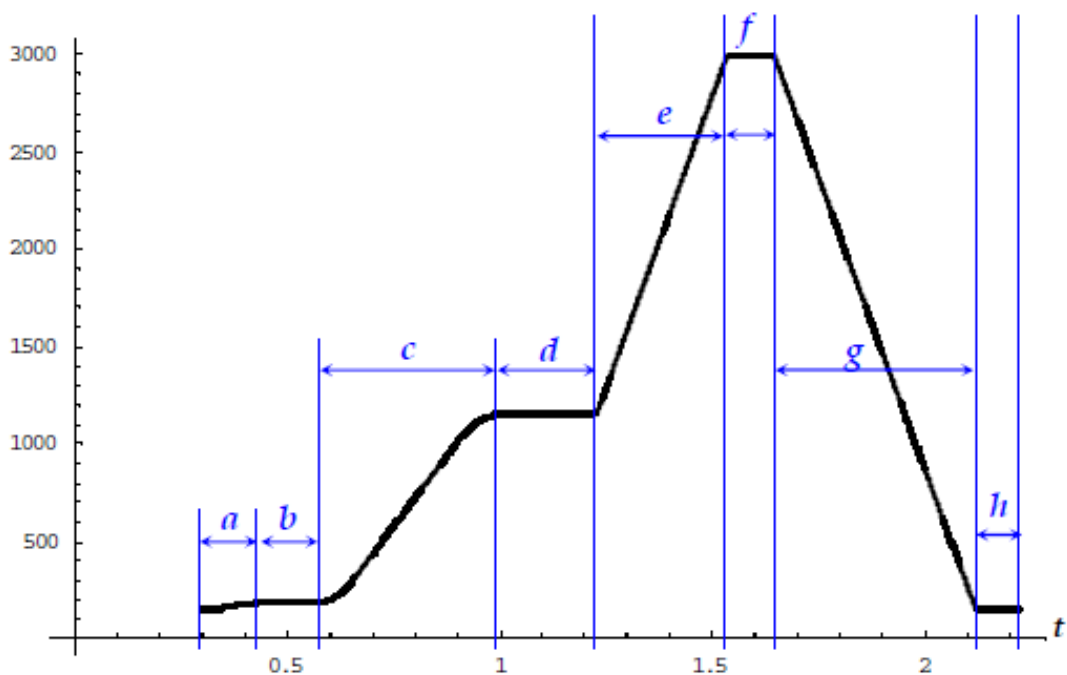


Figure 2.4: General magnetic cycle trend[24]

The cycle begins with the magnet at a current lower than any value used. The first ramp *a* brings the magnets to the injection field; in the second section *b*, called "flat bottom" in the following, the field is constant and equal to the field necessary for the injection. During this period, multi-turn injection, capture of the beam in the RF bucket, scraping and any other preparatory manipulations of the beam take place.

Section *c* corresponds to the acceleration of the beam. The duration of this ramp and its final value, depend on the desired extraction energy for the beam. The curve *c* consists of an initial curved section, followed by a linear ramp and concluded by a second curved section. These curved sections are polynomials of 5th or 7th degree that connect the flat bottom with the linear ramp and the linear ramp with the next flat top, ensuring the continuity of the first and second derivatives (and third for the 7th degree polynomial). In all proton curves, the initial and final polynomials are fixed and from one energy to another only the duration of the linear stretch between them changes. The same occurs with carbon beams, but with different polynomials from those of protons.

Section *d* of the magnetic cycle is the so-called "flat top". During the flat top, the magnets are kept at a constant field corresponding to the desired energy for the extracted beam.

At the end of the extraction, the magnets are brought to maximum excitation and then back to minimum current ready for a new cycle (sections *e*, *f*, *g*, *h*). This is necessary to always follow the same hysteresis cycle and thus maintain the correct correspondence between the current in the feeders and the magnetic field value in the magnets.

Acceleration In the CNAO synchrotron, beam acceleration is provided by a single radio-frequency cavity located in the non-dispersive region opposite the injection and extraction regions.

The action performed by the cavity can be divided into four phases:

- adiabatic beam capture
- acceleration
- beam preparation for extraction
- stabilization of the extracted beam

After multi-turn injection, the beam occupies longitudinally the entire machine. Before it can be accelerated, the beam must be captured by the cavity potential or, as it is called, in the RF bucket (the stable region of the longitudinal phase space). This is accomplished by progressively increasing the gap voltage until the acceptance is large

enough to accommodate the beam. At the end of the process, the beam is packed and ideally there has been no increase in longitudinal emittance. The ratio of the beam emittance to bucket acceptance is called the "filling factor".

After capture is complete, the actual acceleration phase begins. During this phase, the magnetic field of the machine magnets and all the parameters of the cavity vary in absolute synchronism and the beam gains energy at each revolution. The most critical phase is the initial one in which the momentum dispersion increases, the beam length decreases and the space charge effects are larger.

Extraction At the end of the acceleration, the beam must be prepared for extraction. The extraction process is the most critical point of the whole synchrotron project. An extracted beam as uniformly temporally and constant as possible in transverse distribution is a compulsory requirement of a Radiotherapy machine with active distribution.

Ripple reduction All power supplies, including those of dipoles, quadrupoles and machine sextupoles, have some level of residual ripple.⁴

In case the effect of residual ripple on the extracted beam is not minimal, at least for medium-high frequencies, two countermeasures were envisaged at the CNAO:[24]

- the so-called "empty bucket stabilization";
- a feed-forward on the tune with a quadrupole air core.

The first method involves creating an empty bucket between the beam and the resonance. The energy beam pushed by the betatron is forced to turn around this bucket with an effect similar to that obtained by partially obstructing the water barrel: the water comes out with a greater speed. The effect is a reduction in modulation on the extracted beam.

The second method is based on repeatability. A part of the ripple repeats itself cycle after cycle. The idea is to compensate the effect on the tune of the sum of all ripples. This is done by exciting a quadrupole without iron yoke, defined as air core quadrupole, with a sum of breasts with the most visible frequencies in the spill and appropriate phases. The necessary excitation is in fact small but the frequencies can be high and, therefore, it is advisable to remove the iron which is not strictly necessary.

⁴The ripple in electronics is the residual periodic variation of the DC voltage within a power supply which has been derived from an alternating current (AC) source. This ripple is due to incomplete suppression of the alternating waveform after rectification.[25]

Beam energy destruction When the part of the tumor corresponding to a given energy has been completely irradiated, the excess beam must be destroyed. At the end of a given amount of energy, the extraction is stopped and the excess beam is lost during the final part of the magnetic cycle. As it is no longer accelerated when the magnetic field changes its energy, it no longer corresponds to the magnetic field. To avoid spreading losses everywhere inside the vacuum chamber, in a region of maximum dispersion, where the beam moves more due to the error in energy, a dump has been positioned on which the whole beam is lost.

It is necessary to destroy the beam even in emergency cases. An active system has been envisaged in which two fast magnets create a local bump that moves the beam vertically against a dump. The working time of the fast magnets, called dump bumpers, is less than $50 \mu\text{s}$, [24] so as to interrupt the beam in a time corresponding to less than 1% of the dose of a voxel⁵. In order to be able to destroy the beam on the dump and nowhere else, it is necessary that the power supply of the dump bumper continuously knows the stiffness of the beam and maintains the charge of its discharge circuit in accordance with it.

2.2.5. Extraction lines

In the design of the CNAO extraction lines, the strong asymmetry between the vertical plane and the horizontal plane for the extracted beam was taken into account. In the vertical plane, in fact, the beam has a usual appearance, with a bell distribution and elliptical iso-density lines. In the horizontal plane, however, the extracted beam is the section of separator cut by the electrostatic septum.

The approach of the charging bar in the empty ellipse also allows an alternative method for adjusting the horizontal dimension; by varying the progress in phase, the orientation of the charging bar varies and consequently its projection on the x-axis, which corresponds to the horizontal dimension.

The extraction lines can be divided into three parts:

- The line section between the electrostatic septum and the thin magnetic septum inside the ring.
- The initial line section, common to all lines, dedicated to zeroing the dispersion.
- The final section that leads the beam to the various rooms.

The section of line between the electrostatic septum and the thin magnetic septum is

⁵Voxel stands for "Volume Pixel"

internal to the synchrotron and the adjustments in this part consist of a local bump generated with five machine correctors. These allows to adjust the circulating beam in position and angle to the electrostatic septum, and place the gap between the circulating beam and the extracted beam on the thin magnetic septum.

The first external section of the extraction line begins with three magnetic septa, the first thin and the second two a little thicker, fed in series. These septa have the purpose to move away the beam extracted from the synchrotron until it is possible to insert magnets. The first section ends with three dipoles in which the dispersion invariant is zeroed. Interlaced with these three dipoles is a fundamental element of the design: the chopper.

This device, illustrated in Fig.2.5, consists of four dipoles fed in series according to the scheme "+B, -B, -B, +B". When the magnets are turned on, the beam is deflected allowing it to avoid the dump. In this operation, the beam remains on the same trajectory downstream of the chopper and, therefore, remains correctly directed towards the patient.

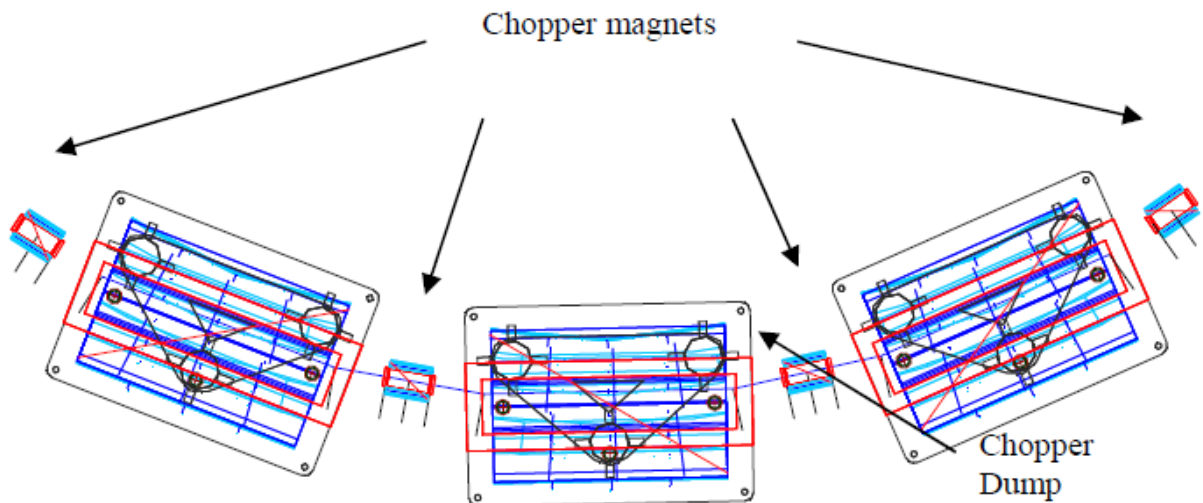


Figure 2.5: Chopper layout[24]

The chopper working time is less than $260 \mu s$ which, considering the ratio between the interval in which the energy beam passes and the interval in which the beam is lost on the dump, allows the beam to be switched on and off in less than 2.5% of the voxel.[24]

2.3. Dose distribution system

The CNAO is designed for a fully active dose delivery system. This means that the tumor is ideally divided into iso-range slices, which are regions that are reached by particles of the same energy. Each slice is then irradiated by brushing it with a thin beam of particles. The concept is illustrated in Fig.2.6.

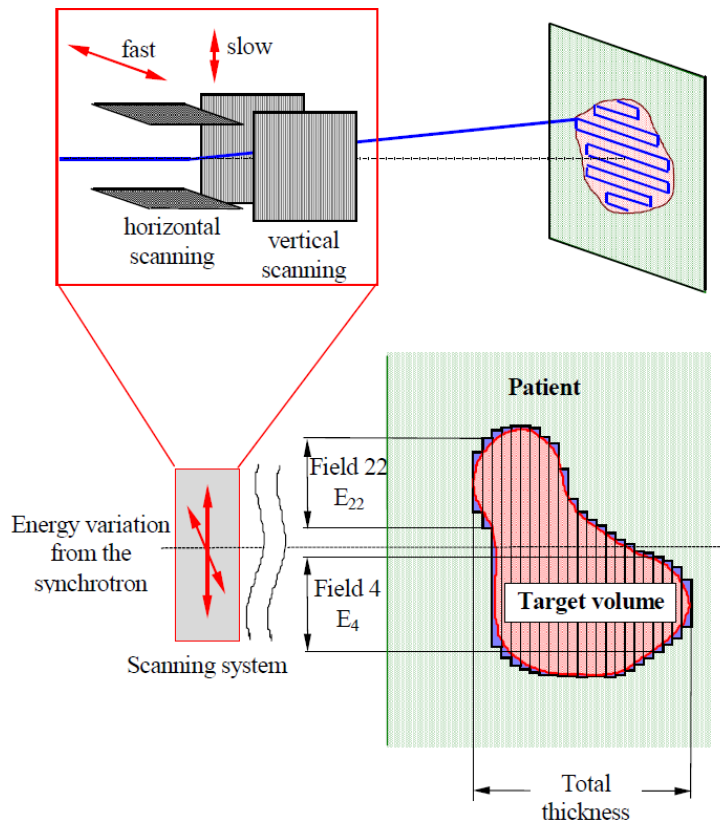


Figure 2.6: Principle of active dose distribution[24]

This way, the beam is directed to the various points of the tumor by varying its position without adding additional thicknesses on the path of the beam that could degrade its quality. The energy is varied by the synchrotron in order to choose the slice and, inside the slice, the beam is moved thanks to two finely laminated magnets, called scanning magnets.

The position of the beam is checked online thanks to a redundant system of monitors that measure in real time the position of the beam and the number of particles received by each voxel. This system is used both to decide when the expected dose for each voxel has been reached and to correct the position of the beam by acting on the scanning magnets.

2.4. Explanation of a cycle

An acceleration cycle is the process of, from a known starting point, injecting particles into the synchrotron accelerator, accelerating the particle beam until extraction, and finally returning back to the starting point. This process, which occurs for every particle beam delivered, is the central focus of the control system for various reasons.

Firstly, the accelerator's subsystems were designed in a way that the vast majority of devices must be provided with their respective configuration settings every cycle before the cycle begins. Likewise, the majority of sensor data is obtained from the accelerator's lower layers only once in the span of a cycle.[26] Tasks that have to be performed in time-span of a fraction of a cycle are relegated to the lowest layers of the control system. Additionally, the synchronization of these tasks is often guaranteed by cycle events generated by the timing system.[27]

An acceleration cycle starts with a specific timing system event. Timing system events contain, among other information, the cycle number and the cycle code that defines, i.a. characteristics, the beam particle and energy. Each particle beam, containing the same cycle code, is by convention referred to as of the same type, mapping to a cycle code every energy level that can be produced by the accelerator.

However, not all cycle codes define a particle beam type. This is due to the fact that empty cycles exist to maintain the accelerator in a stand-by mode. The mapping process, anyway, is an offline configuration process, and not all energies levels that the synchrotron accelerator is able to produce are mapped at all times. On the other hand, a cycle number uniquely identifies every single cycle accelerated since the beginning of the accelerator's operation. Cycle numbers can therefore be used as a time-stamping measurement.[4]

Once the injection process is finished, the start acceleration event synchronizes the acceleration process, accelerating the particle beam to the desired extraction energy. Afterwards, when the energy is met and the extraction event is processed, the particle beam is slowly extracted, leaving the synchrotron. After the extraction, the hysteresis process begins, where the magnetic field in the synchrotron's magnets are set to the maximum value, and brought back to the minimum afterwards. A standard cycle chart is shown in Fig.2.7.[4]

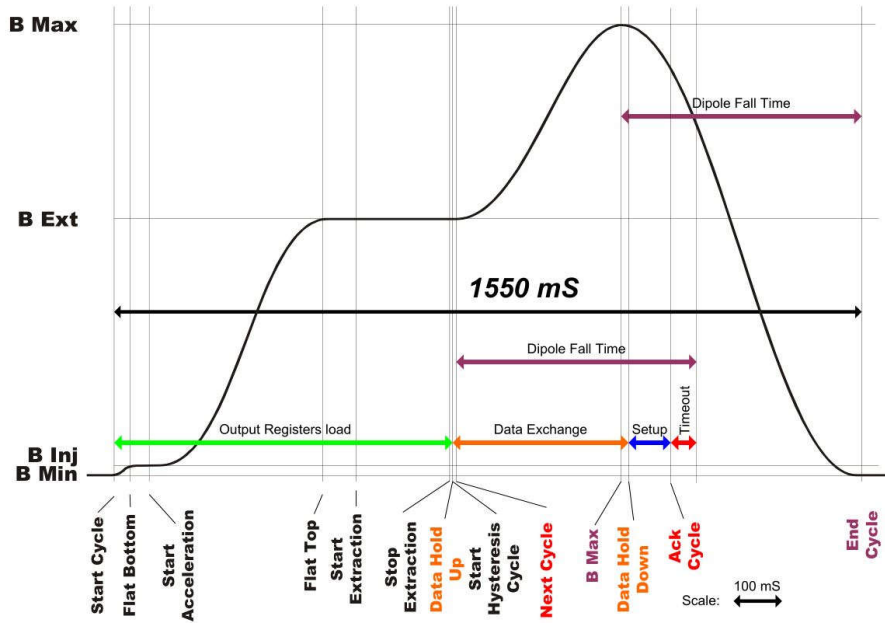


Figure 2.7: Standard cycle chart displaying timing system events[27]

At the CNAO facility, acceleration cycles may have different lengths depending on several factors. The extraction duration, which is variable, has the largest influence on the total length of the cycle.

2.5. Control System General Conceptual Model

The main task of the Control System of the CNAO is to load the equipment with the relevant settings depending on the planned cycles to be executed and to monitor the achievement of the planned results. The operators can supervise the behaviour of the plant by selecting relevant information that is displayed on the operator consoles.

Information can be displayed in terms of screenplays with figures and plots, video signals, oscilloscope signals. Moreover, a general facility allows to receive alarms that inform the operators of some partial unexpected behaviour.

All the settings for each planned treatment have to be kept into a repository that is part of the Control System. It will also keep all the relevant information about the Control System physical design such as number and location of processors and electronic cards, contents of each local computer and distribution of equipment interfaces.

Furthermore, the repository shall also link each patient with his planned treatment in order to keep a record of the medical care and to prepare the plant for the upcoming treatment when the patient is present.

Due to the fact that many settings have to be assigned in a very strict time window, depending on the cycle evolution, the Control System will also supply different time depending on the signals necessary to the plant for triggering activities.

2.5.1. Boundaries and layers

Generally speaking, Control Systems can also include sensors and actuators to acquire information and apply settings. In case of the CNAO project, due to the complexity of the involved equipment, constructing and commissioning sensors and actuators are tasks of different groups. In these conditions, detailed interface specifications have to be established along with the equipment groups, in order to couple sensors and actuators to the Control System. Fig.2.8 shows the main boundaries of the Control System domain.

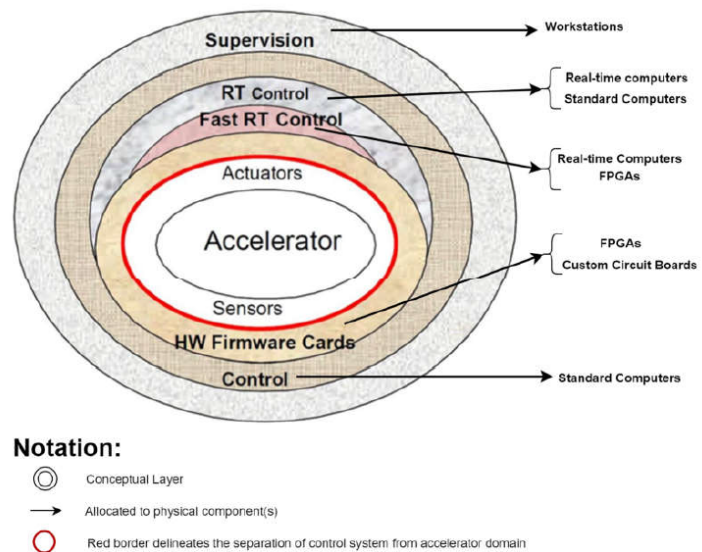


Figure 2.8: The main Control System Layers[28]

The outmost boundary, the supervision layer, is the layer that allows interaction with the users. Users can be operators and in some cases other systems, when the need arises to collect information on treatments for general purposes and when treatments settings have to be loaded into the system for preparing new sessions.

The second layer deals with the abstraction of the equipment in such a way that the upper layer can connect to the accelerator devices without any need to know hardware interface details and using rather general controlling and acquisition concepts such as on, off, control value, acquisition value. The details of this layer will be the subject of the CSRSDs⁶ in which all the interaction concepts will be introduced and fully explained.

The RT-layer will deal with main settings on cycle basis. At given times during the cycle evolution settings will be assigned to actuators and values will be acquired from sensors through the HW-layer. This layer shall not transfer data at a very high rate with the HW-layer, but shall allow to collect and assign values at a maximum of 1 Hz rate; each

⁶The CSRSDs are the Control System Requirements Specification Documents.

processor can manage and transfer about one thousand values at the given frequency.[28] Due to the fact that many more variables have to be accessed, many processors in parallel have to be foreseen to fully service this layer. The processes that manage the operations at this layer will be the subject of a part of the Procedure Document (PROD).

When the need of much faster speed arises, a second deeper layer for RT can be created. In this case, in general, all the information needed for running the equipment is in some way already present into the layer and, from the upper level, in due time during the cycle, comes the indication of which set of data is to be used for the cycle.

Finally, the hardware layer is composed of electronic cards with firmware on board to access the information coming from and going to actuators and sensors.

2.5.2. Component Architecture

The whole Control System is divided into subsystems, which contain parts that are logically or physically linked or belong to a single domain of interest. Then, each of them is further decomposed into Equipment Components, which contain instances that are actually the recognisable logical or physical part of the accelerator.

The characteristics of Equipment Components instances are described and accessed through properties and operations, which do not depend on the particular instance of an Equipment Component, but act on the Equipment Component as a whole.

Each property can be further specified by means of property attributes such as scaling factors, timestamps or beamstamps that describe the context of the information. Meanwhile, each single Equipment Component can be accessed by the external world through a Human Machine Interface (HMI) called Virtual Instrument, that presents to the user all the properties and operations of the specific Equipment Component.

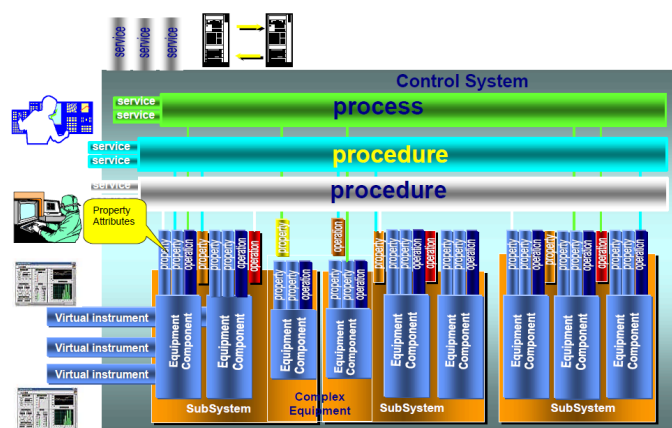


Figure 2.9: Control System Architecture[28]

The Conceptual Architecture of the Control System is shown in Fig.2.9.

Finally, the Control System as a single entity can supply services to other systems in order

to exchange information, such as treatment loading, result acquisition for off line analysis or treatment results to be collected in the patient electronic record.

2.5.3. Control System Layers Architecture

Even if no technical solution would be imposed, a reference architecture for the full definition of the control system needs to be outlined. The layers presented in the Sec. 2.5.1 are translated into a layered architecture based on a network of dispersed processors. Each layer has a set of specific tasks to be accomplished to supply services to the immediately upper level, as shown in the adapted scheme of the architecture in Fig.2.10.

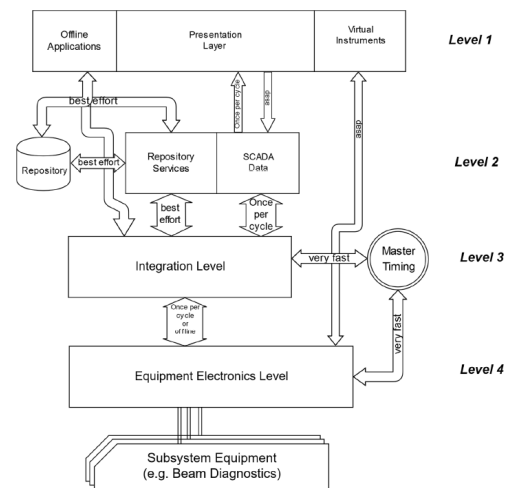


Figure 2.10: Adapted conceptual architecture[29]

The real Control System Layered Architecture is shown in Appendix B.1

The Presentation and Operation Layer The Presentation and Operation Layer is composed of:[28]

- the Human Machine Interface;
- the System to System Interface;
- the Procedures which allow to run the accelerator as a whole to execute the treatments, set up the machine, perform maintenance and enhancements;
- the Alarm Interface that is a special kind of Human Machine Interface treated separately.

Moreover, the repository collects all the information relevant to set up and run the accelerator. A set of dedicated programs allow the entry and maintenance of these data. No configuration data can be kept outside the repository. When needed by the processes, the information is downloaded from the repository into the Concentrator and Data Manager Layer and used to run the machine. When historical data have to be kept for a period of time, they are loaded into the repository from the Concentrator and Data Manager Layer and dedicated programs are used to select and analyse them.

In order to build the user interfaces in the presentation layer, a set of basic components have to be supplied to ease the development, to enable reuse of software and to maintain consistent aspect of the HMI as a whole. Such components are called Visual Components to differentiate them from Equipment Components which act at Concentrator and Equipment Server Layer.

The Concentrator and Data Manager Layer The second level is entitled "Concentrator and Data Management Level". The Control System conceptual model presented in Sec. 2.5.1 is implemented as WinCC SCADA⁷ components. Additionally, the SCADA system in this level is responsible for obtaining data and alarms from the third level and delivered to the upper level SCADA terminals. Operational data from the accelerator is also archived at this level.[4]

This level also contains the repository and the repository services: the repository is a database cluster that contains all information necessary for setting up and running the accelerator. This information includes information on the physical characteristics of the accelerator, configuration settings for software applications, accelerator settings for all planned treatments, and information to link patients to their planned treatments.

Information in the repository is only transferred to the lower levels by the repository services when explicitly directed to do so by maintenance operators, and never in the middle of the operation of the accelerator. Only applications which do not directly interact with the accelerator are able to interface with the repository on a regular basis.

The repository services, along with the repository and support front-end applications, form the Configuration and Supervision Environment.

The Equipment Server Layer The Equipment Server Layer is composed of:[28]

- The RT Processes to transfer to and from the field the relevant data synchronised with predefined time dependent signals distributed by a central facility.
- The Processes which allow the direct transfer of data between the field and the Concentrator and Data Manager Layer. It is strongly recommended that the interface between the Equipment Server Layer and the upper layer are built in a fully standardised way, so that changes in the hardware devices and in the lower level processes would not affect the rest of the Control System.

⁷SIMATIC WinCC is a supervisory control and data acquisition (SCADA) and Human Machine Interface (HMI) system from Siemens. SCADA systems are used to monitor and control physical processes involved in industry and infrastructure on a large scale and over long distances.

- The Electronic Cards which allow the link between the Software contained in the layer and the field.
- The software interface between the processes in the layer and the electronic cards.
- The process to read and store the alarm events and to prepare them to be transferred to the upper levels.
- The software interface between the Equipment Server Layer and the Equipment Electronics Layer.

The Equipment Server Layer can be implemented by means of two distinct approaches.

The first approach realizes the layer by means of a set of crates in which a general purpose microprocessor card fully equipped with extended memory, embedded operating system and relevant software is coupled with specific cards to access the field. The second approach implements the layer by means of off-the-shelf industrial control equipment.

The Equipment Electronics Layer The equipment electronics level is the most diverse, as components in this level are mostly developed by the various groups designing the accelerator systems. This level is the closest to the accelerator equipment, and possesses the strictest real-time requirements.

Components in this level are composed of electronic cards with microprocessors or crates with real time processors and electronic cards. Data required for operation in this level should already be contained in the electronic cards before treatment begins. This happens because, during treatment operation, the fourth level receives information from the timing system, in the form of cycle code, and events. Outside of treatment operation, information may be loaded to the fourth level from the equipment server level.

2.5.4. Control System Synchronisation Architecture

One of the most challenging problems in a control system in which parameters change at a fast rate is to present information in a homogeneous and synchronised way. Each procedure and process has to treat and present information that belongs to the same time slice or event. For this reason a publish and subscribe mechanism is present in the Control System.

Each process and procedure subscribes for the acquisition of information coming from the Equipment Components that, as seen in the previous section, publish their properties and operations. In due time the publish and subscribe mechanism shall supply the processes

and procedures with the information they have asked for, corresponding to the given time slice.⁸

2.5.5. Control System Sequencing Architecture

Each treatment is composed of many cycles. Each cycle has its own characteristics that can be represented by the level of energy of the beam produced in the cycle.

In a treatment, cycles with different cycle types have to be executed in a predefined sequence depending on the cycle type, to achieve the right amount of dose to be supplied to the patient. The position in the sequence of each cycle is represented by its Cycle Number.[28]

During the treatment, some cycles supply less than the theoretical dose to patients, for reasons as patient movement and low quality beam. In such cases cycles have to be repeated and the number of produced cycles does no longer correspond to the sequence Cycle Number.

The number of cycles executed before a given cycle and including it is the Actual Cycle Number. Therefore, the best performance is achieved when the Actual Cycle Number of the last cycle in the treatment is close to the Cycle Number.[28]

In order to set all the parameters of all the equipment at the right value to reach the planned condition, a reliable synchronisation mechanism has to be set in place. Each equipment will be loaded with a set of values representing the individual set points for a predefined number of energy levels. Thus, the number of settings each equipment should be able to keep is the total number corresponding to all possible energy levels the machine can reach.

Once the settings are loaded into the machine an indication of which set has to be used for the next cycle has to be supplied to all the equipment in due time. To obtain this result, an event generation system shall load the sequence foreseen for the treatment and at a given moment during the cycle shall send an event containing the indication of the kind of cycle to be produced next.

Each equipment shall be enabled to receive this event and to decode the cycle type to be produced. The result of the decoding activity is the value or the set of values to be transferred to the actuators in order to reach the desired results.⁹

⁸The typical time slice is the cycle.

⁹Not all equipment settings shall be influenced by the cycle type, thus only the variables showing a dependency on the cycle type will have to adhere to this model.

2.5.6. Control System Communication Architecture

The layers outlined in the previous subsection, when residing on different machines, have to be connected by means of suitable networks. The communication medium connecting the layers shall be a widely available commercial product supporting standard protocols. For suitable security and performance purposes bridging and switching devices shall be installed in order to hierarchically structure the network.

As of now the natural choice is 1 Gigabit Ethernet based communication cards supporting TCP/IP Protocols.[28] Nevertheless such a statement shall not be a constraint.

3 | Medical Devices Certification

The CNAO facility performs Hadrontherapy, hence it is governed by medical regulation. Moreover, the control system software is classified as medical software and it has to comply with medical software standards. All software that has been developed with the intent of incorporation into a medical device is classified as medical device software.

As a consequence, the accelerator control system follows the regulatory standards for medical software. Under the regulatory environment for this software in member states of European Union, certification is accredited on the basis of complying with the development process defined in several accepted standards. These standards are the MDR 2017/745, IEC 62304 and the ISO 14971.

3.1. MDR UE 2017/745: New regulations in the medical field

On 5 May 2017, two new regulations on medical devices were published, and they came into force on 25 May 2017. The Medical Devices Regulation (MDR) date of application is 26 May 2021 meaning compliance is mandatory to be able to place Medical Devices on the European market from this date, unless the transition arrangements allow the continued releasing devices on the market.

Medical Devices Regulation (EU) 2017/745 has replaced the Medical Device Directive (MDD) and the Active Implantable Medical Device Directive (AIMD), whereas the IVDR will replace the In Vitro Diagnostic Directive (IVDD). Both regulations bring a series of important improvements to conformity assessment for medical devices with the intention to:[30]

- improve the quality, safety and reliability of medical devices placed on the European market,
- strengthen transparency of information related to medical devices for consumers and practitioners,

- enhance vigilance and market surveillance of devices in use.

The regulation aims to ensure that, in the European Union market itself, only products that are compliant and meet requirements offering a high level of health protection are available. At the same time, it sets high quality standards for medical devices, in order to meet the common safety requirements relating to these products.

As far as software is concerned, the new regulation addresses the regulation of medical software in a direct, explicit and detailed way. While the old directive mentioned software for medical use only in the very definition of a medical device and to establish that it should be considered as an active medical device, in the new regulation medical software is mentioned several times.

Each medical device, to be put into service, must have the CE conformity marking, shown in Fig.3.1. In order to be able to affix the CE mark, the manufacturer must prepare the technical file, i.e. a set of documents that demonstrate the compliance of the medical device with the CE requirements.[31]

This conformity is assessed by the competent bodies, or Notified Bodies, which examine the documentation and assess its consistency with the provisions of the regulations.

If no non-compliance is found, the Notified Bodies¹ issue the CE mark, which will be affixed to the device in a visible, legible and indelible way.[31]

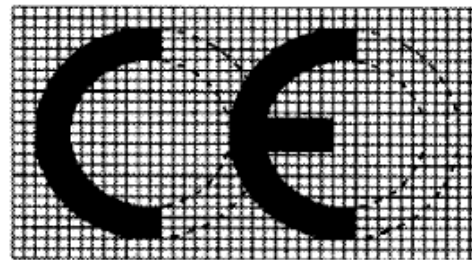


Figure 3.1: CE marking of conformity[31]

3.1.1. Classification criteria for medical devices

According to the regulation, medical devices are classified into four classes: class I, IIa, IIb and class III.[32] Classification occurs according to specific rules, called classification rules.

The rules are divided based on the type of device. Medical software, which are considered medical devices, as they fall within the definition of medical device of the regulation, are considered active devices and therefore can be classified according to rule 11, specifically introduced for medical software.[32]

¹A notified body, in the European Union, is an organisation that has been designated by a member state to assess the conformity of certain products, before being placed on the EU market, with the applicable essential technical requirements. These essential requirements are publicised in European directives or regulations.

The rule states that:

"Software intended to provide information used to make decisions for diagnostic or therapeutic purposes is in class IIa, unless such decisions have effects that could cause:

- the death or irreversible deterioration of a person's health condition, in which case it is in class III, or
- a serious deterioration in a person's health conditions or surgery, in which case it falls into class IIb.

The software intended to monitor physiological processes falls within class IIa, unless it is intended to monitor vital physiological parameters, where the nature of the variations of these parameters is such as to create an immediate danger for the patient; in this case it falls into class IIb. All other software falls into class I".[32]

3.2. Standard UNI IEC 62304: medical device software - life cycle processes

In most cases, software is now an integral part of medical devices. Therefore, it is necessary to ensure that even the use of medical software respects the safety and effectiveness of the device without introducing an unacceptable risk.

The IEC 62304 standard defines software for medical devices as follows: "Software system developed for the purpose of being incorporated into a medical device or which has been developed or intended to be used as a medical device itself".[33]

The standard also establishes a software development process consisting of a certain number of activities, which are shown in Fig.3.2. These activities are, in turn, divided into a set of tasks. The manufacturer must carry out the processes, activities and tasks in such a way as to satisfy the conformity of this standard.

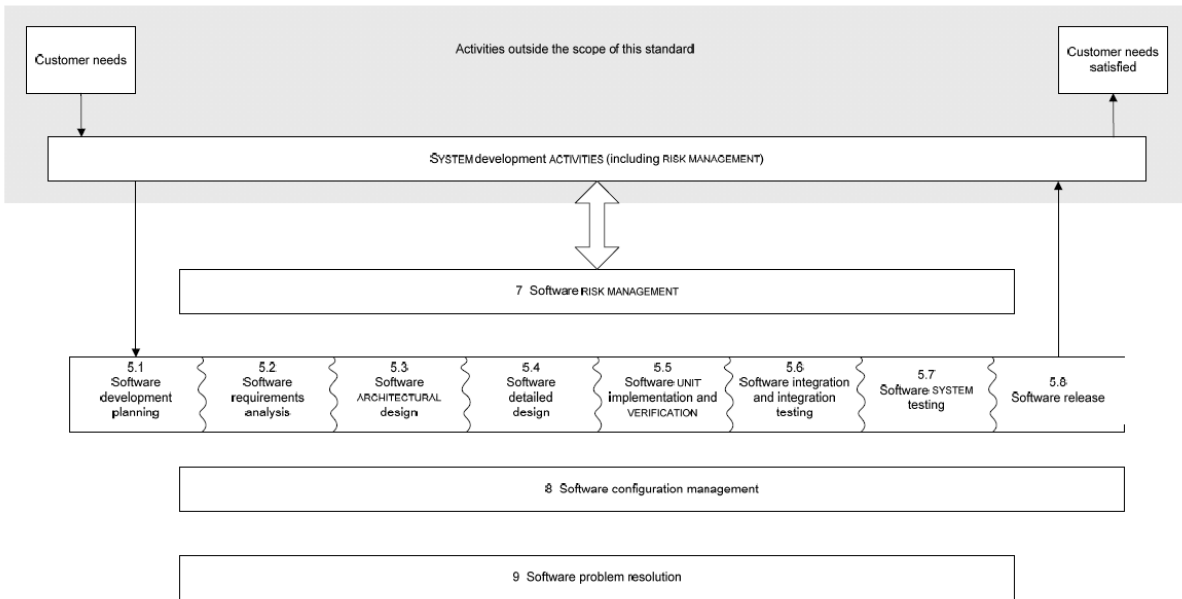


Figure 3.2: Overview of software development processes and activities[34]

Another critical important process is the maintenance process, as many incidents are due to inappropriate software updates and enhancements. The overview of this process is shown in Fig.3.3.

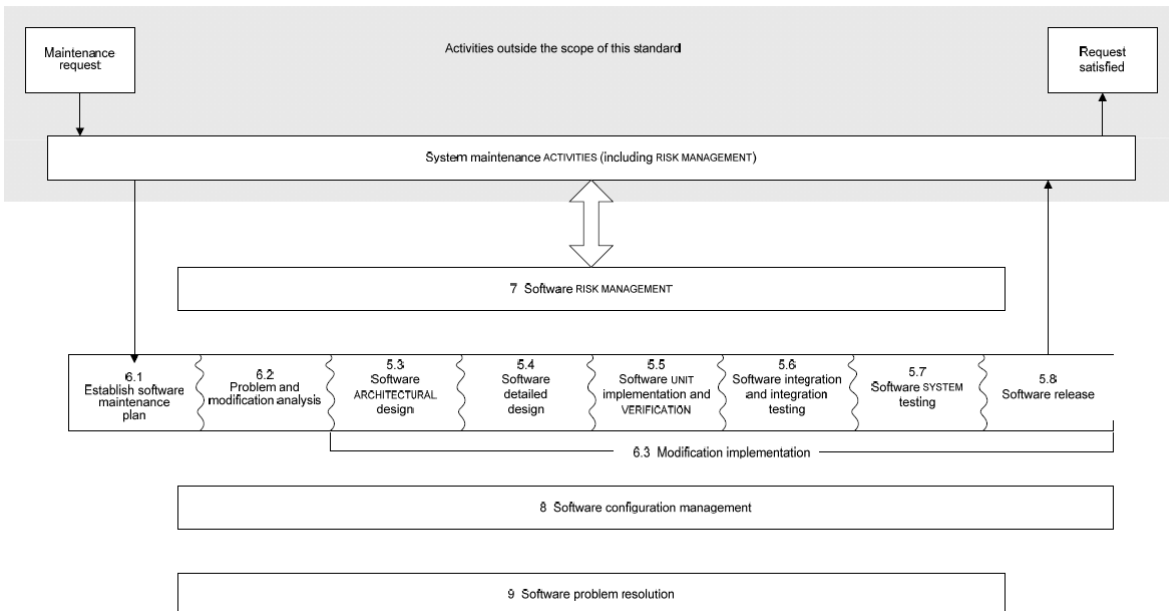


Figure 3.3: Overview of software maintenance processes and activities[34]

This standard indicates two other additional processes: the software configuration management process and the software troubleshooting process.

The first process is to define a scheme that identifies the configuration items and their versions, which make up the software system configuration.

In the second problem-solving process, the manufacturer must analyse the problem and identify the causes, assess the importance of the problem for safety using the software risk management process, document the result of the study and, finally, create requests for changes for the interventions deemed necessary to correct the problem and, if no intervention is possible, document the reasons.

Finally, the risk management process is dealt with in the ISO 14971[35] standard and article 7 of the IEC 62304 standard[36] on medical software describes additional requirements related to software risk management to identify software factors that contribute to the onset of hazards.

The software risk management process must be incorporated into the overall device risk management process. The security class of the software must be indicated in the risk management documentation.

Therefore, the manufacturer must assign to each software system a security class of the software itself based on the possible effects on the patient, operator or other persons, caused by a danger to which the software system may contribute. There are three classes, based on severity, divided into: class A, class B and class C.

The software of class A are such that, following their possible failure, they cannot cause any injury or damage to health; those that belong to class B are those that, given a dangerous condition due to their failure, can cause a minor injury; finally, for software that are in the last class C, their malfunction or failure can cause death or serious injury. If a software failure could lead to a dangerous situation, the probability of failure should be considered at 100%.[37]

By adopting hardware risk control measures or reducing the probability of failure, it is possible to switch from one software security class to a lower one. For example, by reducing the likelihood of death or serious injury following a failure, the safety rating of the software can change from C to B; or if the risk of a minor injury is reduced to such a level that a possible software failure cannot cause any injury then the software can switch from security class B to security A.[38]

3.2.1. Software development process

The software development process, for all three class types, is defined by steps to obtain different requirements. Moreover, the manufacturer must include among the requirements of class B and C software, risk control measures implemented in the software in the event of hardware failures, or in the case of potential defects in the software itself.

The development of the software architecture, which basically consists of a block diagram, represents the structure of the system and a description of the software units conceived as functions or individual modules, thus identifying all the functional blocks of the software. Therefore, the software architecture is necessary to ensure the correct implementation of the software requirements.

The manufacturer must define in detail the software architecture down to a software unit level and, finally, must integrate the software units following an integration plan.

The legislation identifies the three levels of software breakdown: the highest level is the software system, which is made up of one or more software elements and each software element consists of one or more software units. In order to ensure that the integration of the software took place correctly, the manufacturer must test the software elements resulting from the integration of the software units and check if they have the expected behavior. The integration tests that must be carried out include white box tests, also called structural tests, and light box tests, which involve testing the operation of each individual software element and is hence the process of verifying the interaction between software units. The tests are only accurate if the person carrying it out knows exactly what the software element is supposed to do. Therefore, this type of tests require specific knowledge of the internal structure of the software element, such as the source code.[39]

Conversely, software system tests demonstrate compliance with a specific software functionality in the overall system. The software system tests focus on black box tests, also called behavior tests or functional tests which ensure the presence of functional specifications, indicating the part of the specification that has not been met. This type of test is carried out by accessing the software only via the user interface and must be carried out by people who have not participated in the design of the software.

The software system tests verify the behavior of the entire system as a whole. In order to fully test a software product, both tests may be required.

The device manufacturer is responsible for ensuring that the software functions correctly from the point of view of its use. Furthermore, the manufacturer will have to keep track of all the tools used for the development of the software, in order to minimize possible

future misuse.[39]

3.2.2. Software maintenance process

The software maintenance process differs from the development process because if it were urgent to implement changes to solve problems, a shorter process than the complete software development one could be used. Whether the user of the software believes that the actual or potential behavior of the software is unsafe, inappropriate for the intended use or contrary to specifications, the manufacturer activates a surveillance scheme for the collection of data relating to problems in the field, in order to communicate with other users and regulatory authorities.[39]

This process is triggered when the software is subjected to changes in the code or associated documentation due to problems, the need for improvements or adaptations. The maintenance process also includes moving the medical device software to other environments or platforms for which it was not previously released. Finally, it is important to verify that, after the maintenance process, the modified software does not introduce hazards or risks that were not previously present.

3.2.3. Software risk management process

Software risk management is part of the entire risk management of the medical device and cannot be assessed independently from the latter.

This norm provides additional requirements for software risk control, both for the software identified during the risk analysis as potentially capable of contributing to a dangerous situation and for the software used to control the risks of the device itself. The analysis of the dangers of the device provides that dangerous situations are identified and risk control measures are undertaken, often assigned to software functions that implement control measures, in such a way as to reduce the probability and severity of situations to an acceptable level.

The potential causes of the software element that contribute to a dangerous situation and that the manufacturer must identify are:[39][38]

- incorrect or incomplete functionality specifications,
- software defects within the functionality of the indicated software element,
- the failure or unexpected results generated by the SOUP (Software Of Unknown Provenance),
- hardware failures or other software defects that could lead to its unpredictable operation,
- reasonable foreseeable improper use.

The SOUP is the software element previously developed not aiming to be incorporated into the medical device and for which no information is available relating to its development processes. If a failure or unexpected results generated by the SOUP represent a potential cause that may lead to a dangerous situation, the manufacturer must evaluate the lists of anomalies published by the supplier of the SOUP component, used within the medical device. This is common to determine whether some of these anomalies could lead to a sequence of events which, in turn, could lead to a dangerous situation which can be evaluated once the software architecture is finished. In fact, only afterwards it is possible to analyze the associated risks and undertake risk control measures.[38]

3.3. Standard UNI IEC ISO 14971: application of risk management to medical devices

According to this standard, the risk is defined as the combination of the following two elements:[40]

1. the probability of the damage that could occur;
2. the severity of the damage.

The international standard UNI IEC ISO 14971 specifies a process for risk management, through which the dangers associated with the use of the medical device are identified. This may affect both the patient and the operator, but also other people, other equipment and the environment.

In the risk management process, the manufacturer is required to establish, document and maintain a continuous process, throughout the life cycle of the device, with the aim of

identifying the hazards, estimating and assessing the associated risks, controlling these risks and monitoring the effectiveness of the control.

Therefore, the risk management process must be part of the design of the medical device and must include the following elements:

- risk analysis;
- risk evaluation;
- risk control.

A schematic representation of the risk management process is shown in Fig.3.4. Depending on the specific life-cycle phase, individual elements of risk management can have varying emphasis. Also, risk management activities can be performed iteratively or in multiple steps depending on the medical device.[35]

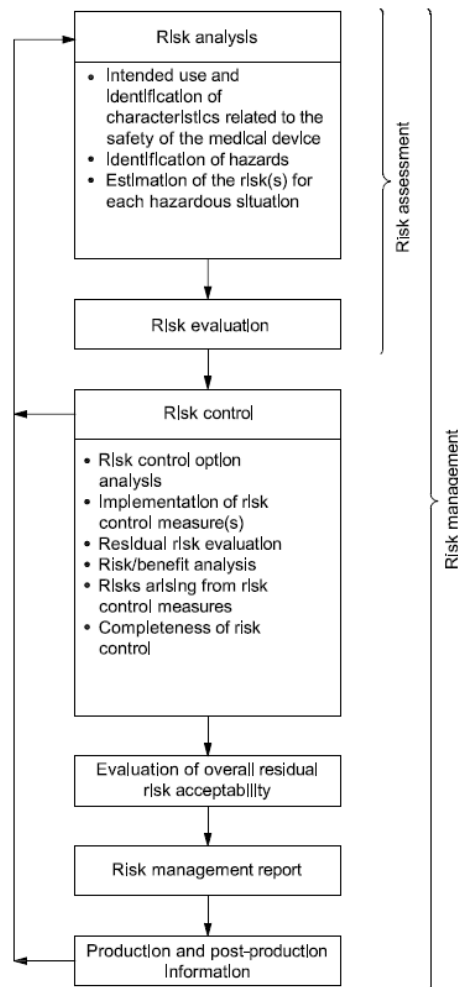


Figure 3.4: A schematic representation of the risk management process[35]

3.3.1. Risk analysis

In the first phase of the risk management process, the manufacturer must be able to identify the characteristics that could affect the safety of the medical device. It is therefore important to consider future use to understand what the risks may be due to the potential uses of the device, as it must be taken into account that the devices could be used in situations other than those envisaged by the manufacturer when the device was initially created.

The manufacturer must herefore document the intended use and any reasonable foreseeable improper use, identify and document all the qualitative and quantitative characteristics that could affect the safety of the device and what are the possible relative limits.

With regard to the identification of foreseeable dangers, it is, on one hand, necessary to consider the dangers associated with the device both in normal conditions and in default conditions. The legislation provides a non-exhaustive list of the hazards that can be associated with different medical devices.

In estimating the risk, on the other hand, all the sequences of events or combinations of reasonable foreseeable events that can occur in a dangerous situation must be considered. Risks may be present both during normal operation conditions and in the event of malfunction.

Thus, the components of risk, probability and the possible effects for both conditions must be analyzed separately. To do this, the manufacturer must use a systematic way to categorize the severity levels and the damage occurring.

According to the definitions, a hazard cannot result in harm until such time as a sequence of events or other circumstances, including normal use, lead to a hazardous situation. At this stage the risk can be assessed by estimating both severity and probability of harm that could result, as shown in Fig.3.5.[40]

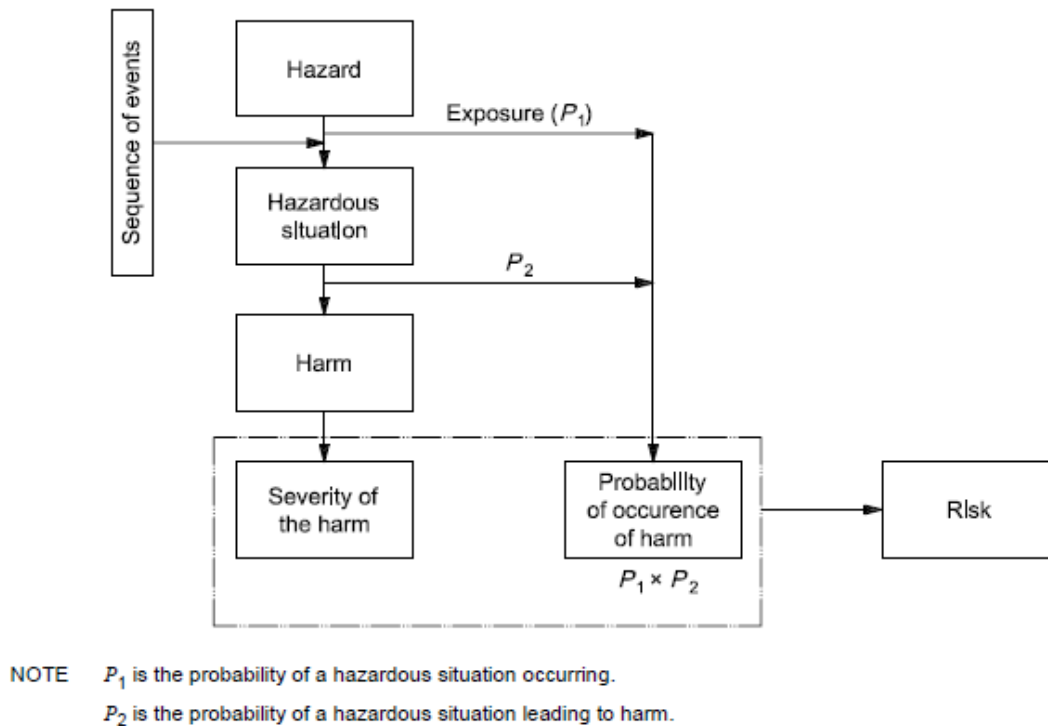


Figure 3.5: Pictorial representation of the relationship of hazard, sequence of events, hazardous situation and harm[41]

This international standard does not require the use of a particular method for estimating risk because it provides a quantitative estimate when sufficient data are available, while it prefers to make a qualitative estimate when such data are not available.[38]

Matrices can be developed to classify risks by studying the two components separately, namely probability and severity. It is mandatory to define N levels of probability, M levels of severity, building a matrix with N rows and M columns. A discrete number of levels are used for probability and at least three levels should be used to facilitate decision making, which can be descriptive or symbolic.[42]

When the accuracy of the estimated probability is in doubt, it is often necessary to establish a wider range for the probability. In the absence of data on the probability of occurring damage, it is necessary to evaluate the risk on the nature of the damage itself: if it is concluded that the danger is of little consequence, the risk is deemed acceptable and risk control measures are not necessary; if, on the other hand, the dangers are significant and therefore could lead to very serious damage, the risk estimate must be made on the basis of the worst case probability estimate.

3.3.2. Risk evaluation

Using the criteria set out in the risk management plan, the manufacturer must decide for each hazardous situation identified whether risk reduction is necessary. This international standard does not specify what the acceptable risk is, as this decision rests solely on the manufacturer. The methods for determining the acceptable risk are as follows:

- Use the applicable standards that specify the requirements which, if implemented, indicate the achievement of acceptance regarding particular types of medical devices or particular risks.
- Compare the evident risk levels for medical devices already in use.
- Evaluate data from clinical trials, especially for new technologies or new intended uses.

In addition, for risk evaluation, the state of the art must be taken into account, which provides information on what is currently accepted as good practice. The methods that could be used to determine the state of the art are, for example:

- standards used for the same or similar devices;
- best practices used in other devices of the same or similar type;
- results of confirmed scientific research.

One way to apply the acceptability criteria is to indicate in a matrix which combinations between the probability of the damage occurring and the severity of the damage can be considered acceptable or unacceptable, as shown in Fig.3.6.

3.3.3. Risk control

In case the risk is considered unacceptable, it is necessary to implement a risk reduction procedure to bring it to an acceptable level. So, control activities must be carried out to reduce the severity of the damage, or to reduce the probability that the damage may occur, or to reduce both.

Once the risk control measures have been applied, if the residual risk is not yet deemed acceptable, it is necessary to perform a benefit-risk analysis to determine if the medical device provides more benefits than harm. In fact, the greater risks can only be justified if outweighed by the expected benefits deriving from the use of the device.

In this analysis, information is collected to determine whether the benefits that the patient

receives from using the medical device outweigh the residual risk; if such evidence does not support the conclusion that the clinical benefits outweigh the residual risk, then the risk remains unacceptable; if instead the benefits of the device outweigh the residual risk, the manufacturer must disclose safety information about the residual risk.

Finally, the manufacturer must perform a review of the risk management process before the device is released for commercial distribution, which must ensure that the risk management plan has been properly implemented, that the overall residual risk is acceptable and that methods are put in place to obtain production and post-production information.[38][43]

3.3.4. Risk Management in CNAO projects

To carry out the analysis of individual risks, a FMEA (Failure Mode and Effects Analysis) approach is used. This analysis methodology evaluates what are the potential failures-risks that the software system may have, estimates the probability of these occurring and the intensity of the damage caused, reporting the data in an Excel table.[38]

Later, during the mitigation and control phase, all the preventive and corrective actions useful to minimize the probability of a specific adverse event, which had been identified among the failure modes, are reported.

The advantage of the FMEA technique is to use both an “inductive” and a “deductive” type approach. In the first case, the possible failures of the individual components are examined and, starting from these, the failure on the entire system is deduced; while in the case of a deductive approach, the failure of the entire system identifies the failures on the individual components.[44]

The FMEA method is a prospective tool that deals with possible events before they occur, i.e. it acts in preventive terms. The manufacturer, as much as possible, must take care of minimizing the probability that a riskful event for the user could occur, reducing it to a value considered acceptable according to an analysis of the risk-benefit ratio.

The next step is the risk evaluation procedure envisaged by the CNAO Foundation. For the estimation and evaluation of the risks, the following indices have been defined:[42]

- Probability Index (P), which indicates the probability that the damage will occur.
- Severity index (G), which indicates the potential severity of the damage.
- Risk index (IR), defined as the product $G \times P$.
- Effectiveness index of technical - design and operational (E) remedies adopted to reduce the risk, must be between 0 and 1 as effectiveness increases.
- Residual Risk Index (IRR) defined as the product of $IR \times (1-E)$.

The Tab.3.1 reports the probability index and the severity index.[38]

Score	Probability Level	Severity Level
1	Rare probability	Low damage
2	Low probability	Minimal damage
3	Moderate probability	Moderate short-term damage
4	High probability	Significant long-term damage
5	Extremely High probability	Permanent damage/death

Table 3.1: Scores on severity and probability levels.[44]

The risk acceptability matrix shows a possible relationship between the probability of occurrence of a failure and the intensity (severity index) of the associated harmful effect, highlighting the following risk regions:[38]

- negligible (green area)
- low (yellow area)
- medium (orange area)
- unacceptable (red area)

CNAO Foundation has defined different bands, as reported in Fig.3.6 and in Fig.3.7.

			Damage severity levels (G)				
			Negligible	Low	Medium	High	Extremely High
			1	2	3	4	5
Probability levels (P)	Extremely High	5	5	10	15	20	25
	High	4	4	8	12	16	20
	Medium	3	3	6	9	12	15
	Low	2	2	4	6	8	10
	Rare	1	1	2	3	4	5

Figure 3.6: Acceptability matrix

Risk Index: IR = G x P		
Negligible	1-4	
Low	5-8	
Medium	9-12	
Unacceptable	13-25	

Figure 3.7: Index Risk Evaluation

The CNAO Foundation has decided to accept negligible and low risks, has identified the actions to be taken to manage medium risks and has decided to not tolerate unacceptable risks. [38]

3.4. The Technical File

At the end of all the processes described in the previous sections, the manufacturer of medical devices has the obligation to create a collection of documents and information that describe in detail the device under consideration and in particular the aspects related to the safety and efficacy of the product: from the design phase of the device up to the post-market surveillance. This collection is called "technical file", a document that will be examined by the Notified Bodies responsible for issuing the CE certification, which is essential for the use and marketing of the device.

One thing to take into account is that the technical file is not identical to a design dossier, which can be seen as slightly more in-depth or advanced than a technical file. In the EU, the design dossier is used for the higher risk medical devices.

The EU MDR states that medical device manufacturers must:[45]

- Prepare technical documentation before placing a product on the market.
- Ensure technical documentation is made available to the market surveillance authorities (should they request to see it) as soon as the product is released on the market.
- Keep records of technical documentation for 10 years after the launch date(unless explicitly specified otherwise).

Technical documentation should, at least, have:[45]

- A device description and specification section. This should also have your unique device identification (UDI) number.
- Labeling and instructions for use.
- Detailed information on design and manufacturing. It is mandatory to show manufacturing process, suppliers, and materials used.
- Detailed risk management information in compliance with ISO 14971.[35]
- General Safety and Performance Requirements (GSPR), formerly known as essential requirements. It identifies the obliged actions for the device type. From a design control perspective, the contents of the traceability matrix will assist the user with addressing the criteria of GSPR.

- Verification and validation information. In terms of verification and validation, the European Commission places a heavy emphasis on clinical data - not just during design and development, but post-market, as well. CERs (Clinical Evaluation Reports) should provide a comprehensive overview of the device's design and composition, as well as its intended applications and any relevant literature reviews.
- Post-market surveillance (PMS) information, including PMS plan, post-market clinical follow-up (PMCF) plan, and periodic safety update report (PSUR).

Lastly, the technical file must always be updated and available at any time to be inspected by the competent bodies.

4 | Integrated development environment and *Agile* development approach

As reported in the previous chapters, the operation of particle therapy facilities requires complex control and supervision systems, often spanning several software development environments, each containing many applications.

While the component-based development approach brings many benefits, the integration of the parts is still left to the initiative of each developer without a repeatable path that can be demonstrated when the artifacts must undergo the certification process. Besides, the development of such control and supervision systems is no longer limited to a network based on fixed workstations, but mobile devices must be considered introducing an extra need for enhanced security.

However, existing general-purpose development environments and methods have important shortcomings for the development of such systems, namely:

1. the difficulty to involve domain experts in the process,
2. the difficulty to assure the proper integration of the heterogeneous parts that make up such systems,
3. the difficulty to find development methods that conciliate the needs for fast delivery and stringent quality assurance.

Moreover, because the plant activity affects human beings, building applications to control and supervise such target must undergo a set of quality assurance actions that have the aim to demonstrate the consistency and the reliability of the artifact; additionally, evidence of the accelerator's correct behavior must be provided to the regulatory body.

Much stricter rules have been adopted by the regulatory body since the first construction of CNAO. Because the number of applications necessary to run the facility is quite high,

the amount of work for upgrading or reproducing the center would be devastating without the help of an environment that can demonstrate and document the job being done.

4.1. CF2020 and the Cyber-Physical Systems

To overcome the shortcomings of the development environments, the objective of the CF2020 (CNAO Framework 2020s) project is to create an integrated development environment supporting an *Agile*¹ development approach capable to guarantee very strict quality requirements of complex *Mission Critical* systems for specific domains, in particular, healthcare systems, including:

- tools, based on domain-specific languages (DSLs), to assist domain experts and developers in specifying and maintaining requirements, design decisions, test cases, workflows, state machines, exceptional conditions, and other relevant information at a high level of abstraction, following a document-centric interaction style with domain experts for greater accessibility,
- tools to generate part of the implementation-level artifacts (executable code, configuration information, etc.) from the high-level specifications, to speed up development and ensure consistency between implementation and specification,
- run-time frameworks, including a software bus and a fast workflow management system that effectively realize a reference architecture for the target domain ensuring system integration and the satisfaction of stringent quality requirements,
- an extensive set of execution support libraries that implement recurrent tasks for the target domain, so that the amount of software to be manually built by IDE² users is limited to the specific business logic,
- a set of application models and templates based on the frameworks and libraries, to grant homogeneous and more robust development through the reuse of semi-finished parts,

¹*Agile* is an iterative approach to project management and software development that helps teams deliver value to their customers faster. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.[46]

²An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. Integrated development environments are designed to maximize programmer productivity by providing compact components with similar user interfaces. IDEs present a single program in which all development is done. This program typically provides many features for authoring, modifying, compiling, deploying and debugging software.

- tools to produce documentation, mainly automatic, with the guarantee of strict alignment with the artifacts³,
- tools to produce test specifications to cover the full set of use cases provided by the applications.

The objectives, the real breakthrough, and the methodology of the project are better understood with a clear model of the medical environment in which the medical devices shall operate.

A complex medical device is supposed to have some or all the following characteristics.

- The whole device is the collection of a large set of simpler devices that collaborate to deliver the intended service.
- The device has a complex functioning structure that needs fast control and remote supervision.
- Large sets of configuration information that shall satisfy all the possible execution paths must be made available and distributed.
- The activity of all composing equipment must be synchronized and sequenced using a flexible workflow manager that shall act at any level from the shop floor up to the management dashboards with an extremely large granularity of time frames.
- The device must be linked to the rest of the medical environment to receive the actual running configurations and to deliver information about completion statistics, times, executed jobs, and performance results.
- The device is a very valuable item that needs to be used with the highest usage rate and thus needs very careful dynamic planning support and must be able to return all the information needed to evaluate the actual costs of the resources involved.
- The national healthcare system needs to collect information about device usage and performance.

Thus, a complex medical device is very similar to a production line and, therefore, this work is fully applicable to any production plant in which the need for strict quality requirements and high software/hardware integration is mandatory.

³This specific requirement is the most important part of the project discussed in the thesis. More important details will be discussed in Chap.5.

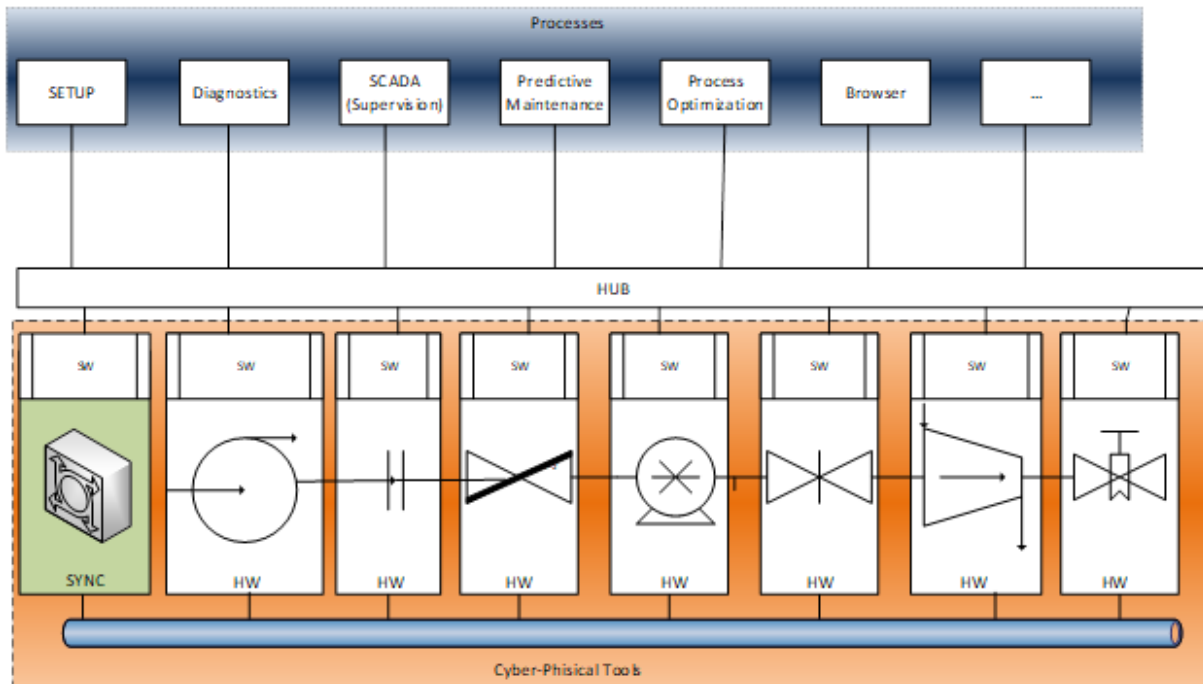


Figure 4.1: Cyber-Physical System (CPS)[47]

The toolkit is an effort for helping industries to adhere to the industry 4.0 paradigm⁴ and to successfully produce Cyber-Physical Systems (CPS), shown in Fig.4.1.⁵[47]

4.2. Product Line production process

A software Product Line is a set of software-intensive systems sharing a common, managed set of features that satisfy specific needs of a particular market or mission, and that are developed from a common set of core assets in a prescribed way.[48]

The comprehensive nature of the Product Line strategy makes it an umbrella under which a range of techniques and methods can be assembled. *Agile* development methods, domain analysis, model-driven architectures, and generative programming can all be part of a successful product line organization.

The Fig.4.2 shows a scheme, carried out by the author, that represents how to build a Product Line Control System.

⁴Industry 4.0 is the digital transformation of manufacturing/production and related industries and value creation processes.

⁵In the Fig.4.1 SYNC stands for Data Synchronization that is the ongoing process of synchronizing data between two or more devices and updating changes automatically between them to maintain consistency within systems. Moreover, HW and SW are the Hardware and Software part of the systems.

Lastly, in a computer network, a device that connects various clients to the server, collecting cables from different computers is called HUB.

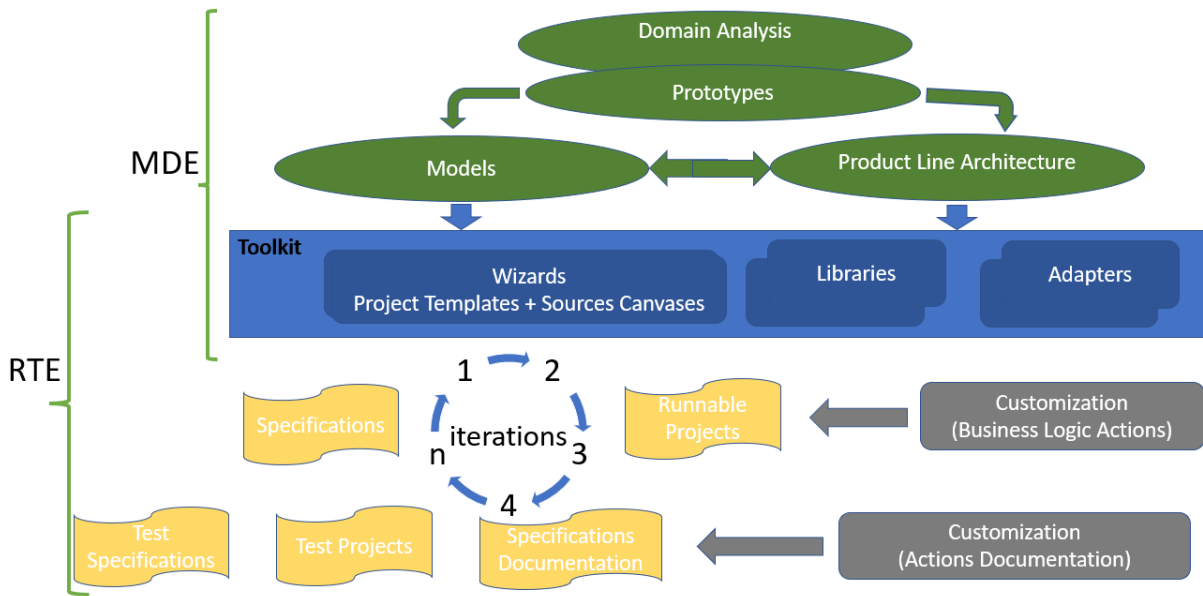


Figure 4.2: Scheme to build a Product Line Control System

Many models can be used to describe a software product line.⁶ Besides, a Meta-Model defines the structure of a modeling language, as shown in Fig.4.3.

Two of the models used in the Product Line Control System are the MDE and the RTE, which will be discussed soon after.

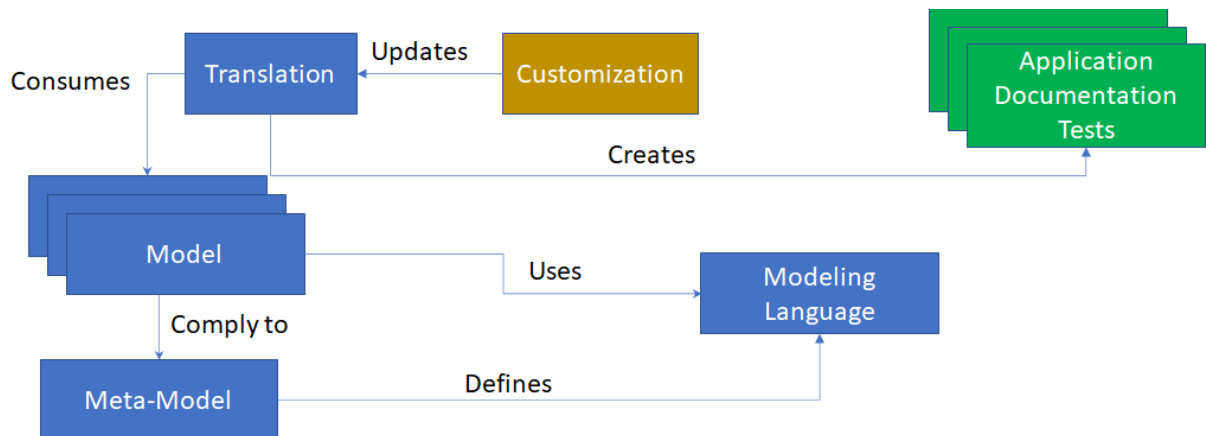


Figure 4.3: Model & Meta-Model

MDE: Model Driven Engineering The MDE approach is meant to increase productivity by maximizing compatibility between systems (via reuse of standardized models),

⁶A model is an abstraction of a system often used to replace the system under study. In general a model represents a partial and simplified view of a system, so, the creation of multiple models is usually necessary to better represent and understand the system under study.

simplifying the process of design (via models of recurring design patterns in the application domain), and promoting communication between individuals and teams working on the system (via a standardization of the terminology and the best practices used in the application domain).

A modeling paradigm for MDE is considered effective if its models make sense from the point of view of a user that is familiar with the domain, and if they can serve as a basis for implementing systems. The models are developed through extensive communication among product managers, designers, developers and users of the application domain. As the models approach completion, they enable the development of software and systems.

MDE is iteratively applied to produce running applications and related documentation and test.

RTE: Round Trip Engineering The Round-trip engineering (RTE) is a functionality of software development tools that synchronizes two or more related software artifacts, such as, source code, models, configuration files, and even documentation. The need for round-trip engineering arises when the same information is present in multiple artifacts and therefore an inconsistency may occur if not all artifacts are consistently updated to reflect a given change. For example, some piece of information was added to/changed in only one artifact and, as a result, it became missing in/inconsistent with the other artifacts.

In order to achieve a sustainable RTE a well defined set of extensions have to be specified in the meta-model. A scheme of the working path of the RTE is represented in Fig.4.4.

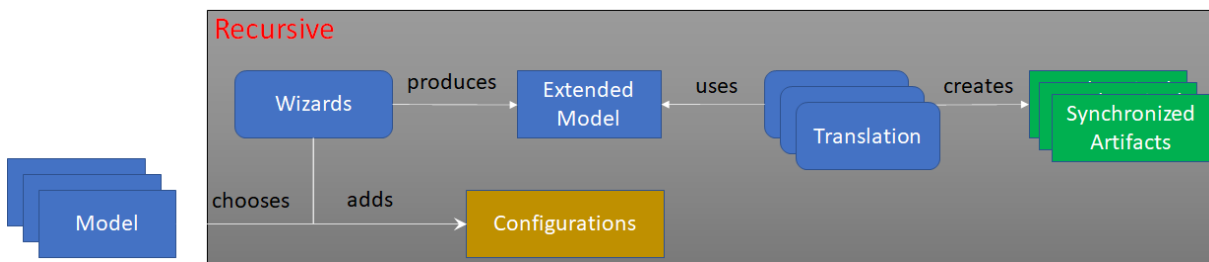


Figure 4.4: RTE scheme

It is often wrongly defined as simply supporting both forward and reverse engineering. In fact, the key characteristic of round-trip engineering that distinguishes it from forward and reverse engineering is the ability to synchronize existing artifacts that evolved concurrently by incrementally updating each artifact to reflect changes made to the other artifacts. Furthermore, forward engineering can be seen as a special instance of RTE in which

only the specification is present and reverse engineering can be seen as a special instance of RTE in which only the software is present. Many re-engineering activities can also be understood as RTE when the software is updated to reflect changes made to the previously reverse engineered specification.

Another characteristic of Round-trip engineering is the automatic update of the artifacts in response to automatically detected inconsistencies. The automatic update can be either instantaneous or on-demand. In instantaneous RTE, all related artifacts are immediately updated after each change made to one of them. In on-demand RTE, authors of the artifacts may concurrently evolve the artifacts (even in a distributed setting) and at some point choose to execute matching to identify inconsistencies and choose to propagate some of them and reconcile potential conflicts.

Besides, the Round-trip engineering supports an iterative development process. After the model is synchronized with a revised code, the engineer is still free to choose the best way to work and make further modifications to the code or make changes to the model. Moreover, the engineer can synchronize in either direction at any time and it can be possible to repeat the cycle as many times as necessary.

4.2.1. Product Line Architecture

The Product Line Architecture is an early and prominent member in the collection of core assets. The architecture defines the set of software components that populates the core asset base.

The Product Line Architecture, together with the production plan provides the prescription for how products are built from core assets. The Fig.4.5 shows how a Product Line has evolved over time.

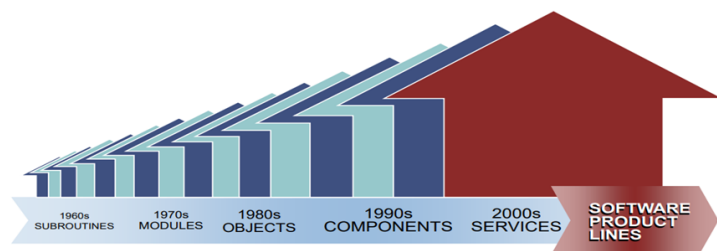


Figure 4.5: Evolution in the Product Line

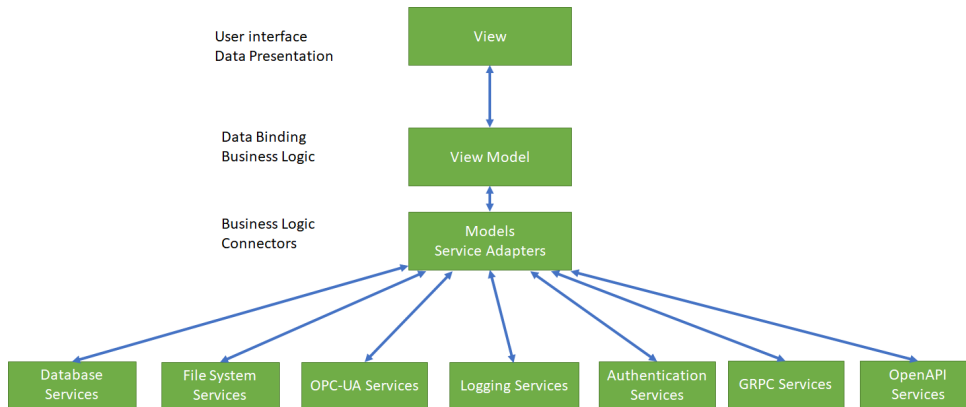


Figure 4.6: NGCS Architecture scheme

In the NGCS⁷ Architecture, the Product Line Architecture defined by the author for the CF2020 toolkit, shown in Fig.4.6,⁸ the customizations of Fig.4.3, and the configurations of Fig.4.4 take the form of Actions, Visual Elements, and Properties. The Action is the only mechanism by which a Model can incorporate the business logic specific to an application. The Visual Elements that are the components of the user interface allow showing and modifying the status of the application and firing the User Actions while the Properties connect Actions and Visual Elements. The Actions and the Visual Elements are specified as Model Configurations so that the translation operation can include them in all the artifacts and are specified by means of Wizards.

Hence, the philosophy of the NGCS Product Line is based on making computers and digital tools work as much as possible.⁹

4.3. Description of the toolkit

As already discussed in Sec. 4.1, the toolkit used in this project is composed of:

- an extensive set of execution support libraries that implement recurrent tasks for the target domain, so that the amount of software to be manually built by IDE users is limited to the specific business logic; the libraries shall also implement a part of the run-time framework,
- a set of application models and templates based on the framework and libraries,

⁷NGCS stands for "Next Generation Control System".

⁸In the Fig.4.6 the OPC-UA stands for "Open Platform Communications Unified Architecture", that is a platform independent service-oriented architecture that integrates all the functionalities of the individual OPC Classic specifications into one extensible framework.

⁹In the NGCS, the products are built from components and services, being validated and documented by the wizards.

granting homogeneous and more robust development through the reuse of semi-finished parts,

- a set of wizards that shall build skeleton applications in which all the ingredients required by the models are included,
- a set of basic services that supply information and allow executing tasks necessary to the models to fulfill their activities. Extra services can be added by developers and the protocol to include them is fully defined,
- a set of tools to produce application documentation based on the same information supplied to generate the running code,
- a set of tools to create test specifications and test cases based on the same information supplied to generate the running code.

The author represents in Fig.4.7 the whole set of elements composing the toolkit.¹⁰

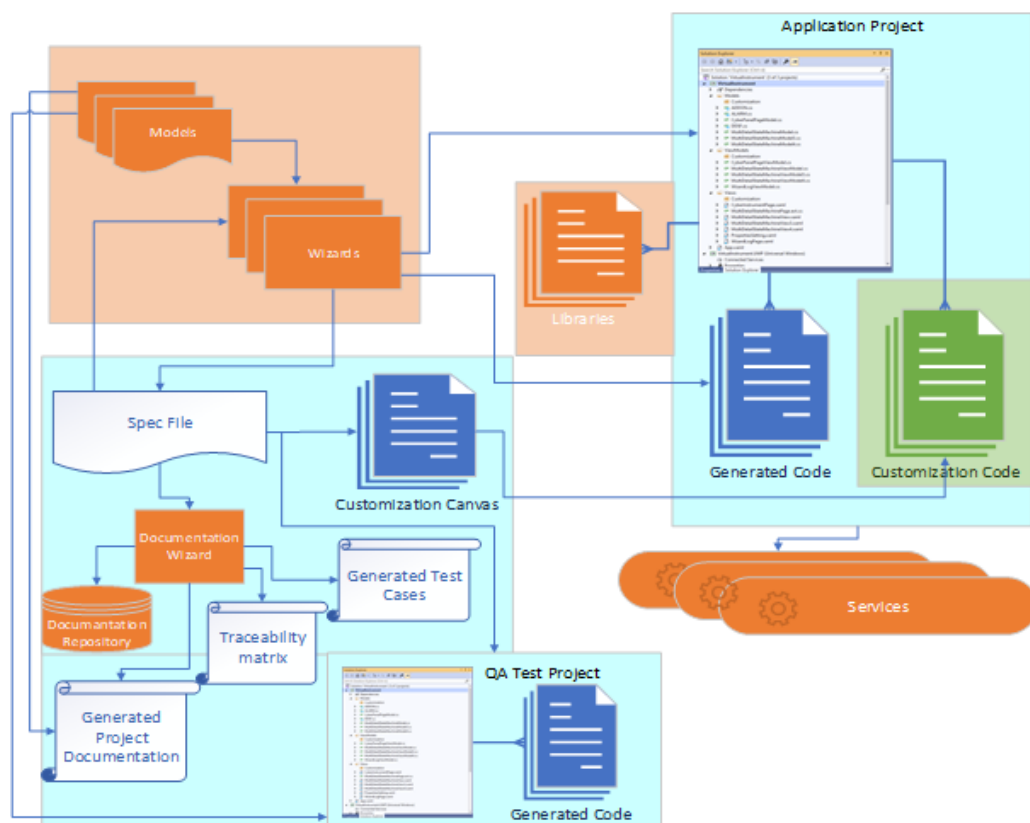


Figure 4.7: Toolkit structure

¹⁰In the Fig.4.7 the Spec. file is the Specification file, that will be discussed in Sec. 4.4.1, while the QA test is the Quality Assurance test, that help web developers to deliver the best possible product within the estimated timeline, detecting any issues that might block the successful operation of the software or even affect the user experience.

The libraries and the wizard produce many advantages such as the reducing of the amount of error made by the developer, in order to allow the developer to concentrate on the business logic; besides, they are useful to reduce the amount of documentation to be produced and enforce the rules and functionalities mandatory in the production environment.

Moreover, in the world of the “Internet of Things”, it is not conceivable to have a production environment without strong support for several operating systems and mobile architecture.[49]

The project shall support the development of applications for desktops and mobile devices and shall include services to access remote databases, remote file systems, and services to control equipment software based on the OPC-UA protocol, services based on GRPC¹¹, and OpenAPI protocols.¹²

4.4. Project focus

Thanks to this system, the author focuses on generating an extended skeleton of the application documentation for different purposes, from the user’s manual to the technical dossier, in order to keep the documentation aligned with the artifacts and enable the constructions of tools that verify and validate the final product for keeping up with the required certification level.

From the scheme of the Toolkit structure reported in Fig.4.7, it can be extrapolated the crucial part of the project that is the realization of a Documentation Construction Path, as shown in Fig.4.8.

¹¹GRPC is a modern open source high performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to connect devices, mobile applications and browsers to backend services.

¹²The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.

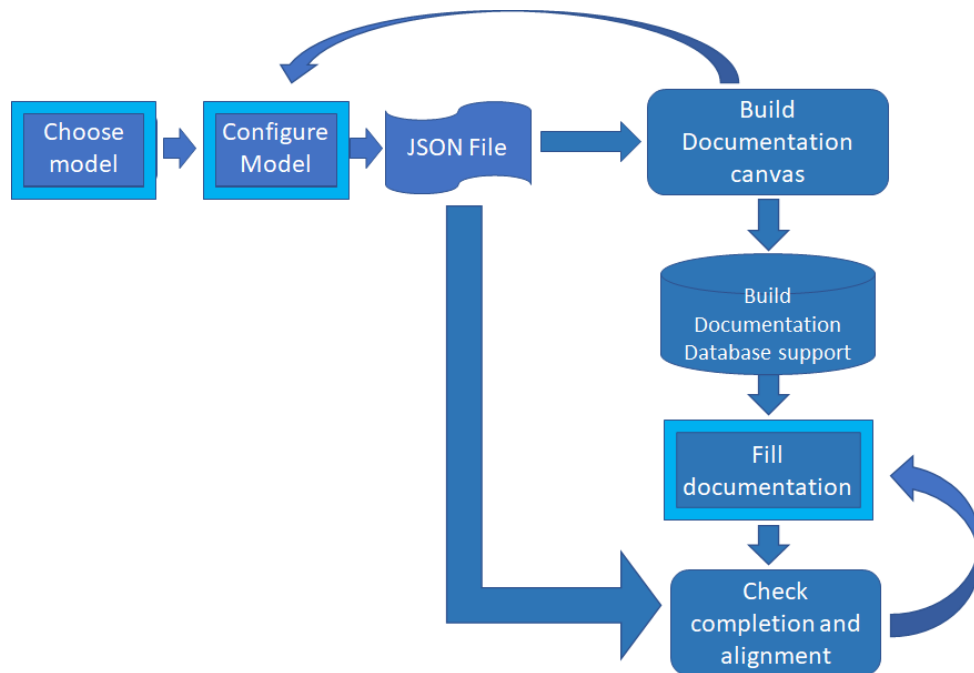


Figure 4.8: Documentation Construction Path

4.4.1. CF2020 Project Wizards

Model-Driven Architecture has been an approach to build applications for a long time. It is an extension of design patterns to encompass the whole application. Models require a deep understanding of the requirements and the commonalities among applications that address a shared domain. Thus, a survey of the entire set of applications presently running in the CNAO Control and Supervision System has been done before, defining and implementing the tools that are contained in the CF2020 toolkit. About 100 applications have been considered.

Models are composed of pages on which data is displayed and managed, a navigation system among the pages, a set of menu items for each page, and possibly a toolbar to execute standard commands such as search and filters. A page having several panels can be configured. Each panel can contain visual elements that show or enable to modify data or to execute extra actions. The models define also the connection to all the services needed to manage the data of the applications.

The project wizards allow the user to select the desired model and the target operating system on which the application will run, to choose the pages that will be included in the final applications, to assign the model configuration parameters, and to define the menu, its items, the toolbar buttons, and the visual elements included in each page.[49]

The types of pages that are included in each model depend on that model. If the pages are enabled to contain detail panels, the developer can add as many panels as needed and possibly choose their layout.

The output of the wizard is a Specification File with JSON¹³ format (JSON Spec. File) that can be used to build a skeleton of an application that is already a running executable. Consequently, the developer knows that all the connections and interfaces are respected and the basic behavior supplied by the chosen model is fulfilled.

The skeleton contains ‘hooks’ to which the developer can attach the code specific to the business logic of the application.

The wizard is not a one-way application.[49] The developer can later modify the project by adding pages, detail panels, menu items, toolbar buttons, and visual elements. Additionally, an undo mechanism has been developed, allowing developers to return the generated application to a previous version of the project whenever required.

A mechanism of hot-reload is available to modify the application user interface while it is running automatically connecting the new elements to the services included in the model. The development kit can be extended by adding new models and new wizards when the target environment requires new types of applications.

A set of test cases validating the common toolkit behavior is supplied. These tests shall be always executed when a new application is built to verify that for any reason the basic functionality has not been disrupted. Then the procedure already devised in CNAO that foresees the production of a test plan, test specifications, and test reports shall be adopted.

The JSON Specification File can be further used to build a skeleton of the application documentation, including already all the choices the developer has done in the wizard.

Finally, it can be accessed to develop a Test Specification File and test cases, because this file already contains all the definitions of the actions that the application can execute.

¹³JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays. It is a common data format with diverse uses in electronic data interchange, including that of web applications with servers.

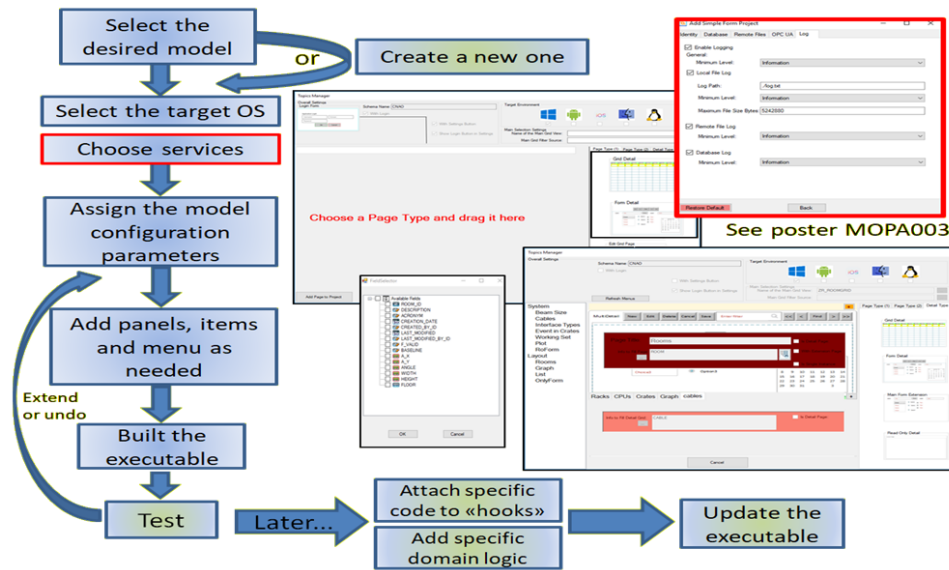


Figure 4.9: Process map without JSON Specification File

The author wants to make a comparison between the two processes, in order to highlight the importance of the work carried out by the author during the internship. In the Fig.4.9 is depicted the process without the JSON Specification File, while the Fig.4.10 shows the complete map of the process highlighting the importance of the JSON Specification File.¹⁴

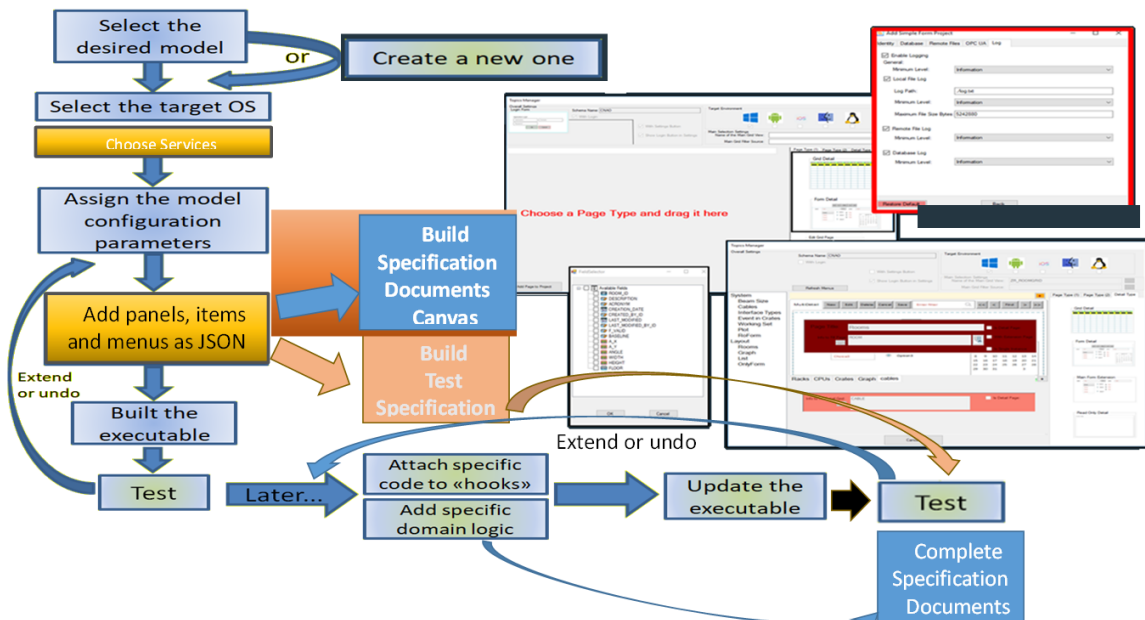


Figure 4.10: Complete process map

¹⁴In the Fig.4.9 and Fig.4.10, OS stands for "Operating Systems" supported by the toolkit.

The wizards have been grouped into two main blocks. Two distinct programs have been built for each block. The first block, named "Build Specification Document Canvas", contains wizards that are targeted to build applications used to manage data contained in a database. The second block, named "Build Test Specification", contains wizards that are targeted to build applications aimed to supervise an industrial plant or a production line such as the CNAO accelerator complex. The first block, displayed in Fig.4.10 and explained in Chap. 5, is the fundamental outcome of the research of the author carried out in this thesis.

4.4.2. JSON Specification File

The applications built by the toolkit can address several operating systems and they can access several databases when they are customized by the user, but the models can target a single database.

The most extended model can contain an arbitrary number of pages that can be accessed by a left side menu¹⁵.

Each page can be a single page, a tabbed page with an arbitrary number of detail pages or a page with a master and an arbitrary number of detail pages.

Moreover, each page can be composed of a maximum of twelve panes depending on the page type. It can be connected to an arbitrary number of database tables depending on the page type, to an arbitrary number of OPC-UA Servers, to an arbitrary number of GRPC based services or to an arbitrary number of OpenAPI based services.

Moreover, each page can have an arbitrary number of menus, which can have an arbitrary number of menu items. Besides, each page can have an arbitrary number of toolbar buttons.

Therefore, there can be the following ancillary optional pages depending on the model:

- a login page,
- a settings page that should contain elements to configure the application (options for colors, optional confirmation for specific actions, ...),
- a setup page that should contain elements to configure the actions performed by the application (number of retries, relevant addresses, ...),
- an extension page for each page in the model to show ancillary data that cannot

¹⁵This menu is named in a technical jargon "Hamburger menu".

find place on the parent page.

When customizing, the developer can use packages that are not included in the model projects. These packages can be declared by the developer while creating the application specifications and are kept in the specification files.

Hereafter, the developer can add application-level resources which are kept in the Specification File and included into the project source files, when building the application.

Thus, in order to contain all the information necessary to build the most extended model the JSON Specification File has got a specific structure defined as a JSON Schema.¹⁶[50]

¹⁶JSON Schema is a vocabulary that allows to annotate and validate JSON documents. Further details about the JSON Schema are shown in Appendix D.

5 | Development Software Validation Documentation

In this chapter, the steps to be taken to validate a general software with the applications created during the period of the internship in the CNAO foundation are shown. Therefore, the description of every single step is reported, from the creation of the application to the release of the software.

In the first step, the author uses a new State Machine (a new wizard that has new applications) in order to create, at the end of the process, a Specification File JSON, containing all the information needed to build the test specifications, all the documentation and, eventually, special cases of interface customization. An important part of the work done by the author is the creation of the test cases and their execution (test run).

The documents for product validation will be generated, in order to create a final documentation for all documents generated and to carry out the necessary checks. It is based on an iterative approach to verify that all documentation is consistent with the work performed.

Finally, the software requirements or functional requirements of the system are outlined by a specific software. In the last phase of the work, it is necessary to answer a series of questions, or to answer a sort of checklist in order to proceed, therefore, with the release of the software.

The drafting of the requirements are facilitated by the use of the Jira software, made available by the CNAO Foundation, which allows to comply as a database of the specifications description with all the mandatory points of validation of a software, according to IEC 62304. The Jira software is a generic software management, which is used by the CNAO Foundation to plan, track and release the software. The CNAO technicians have defined and set up the various steps of the release on the Jira platform, in compliance with the regulations.

5.1. Wizards Application

The block of wizards aimed to build data management systems is grouped in a program named Project Designer. When the program is launched, the dialog shown in Fig.5.1 is displayed.

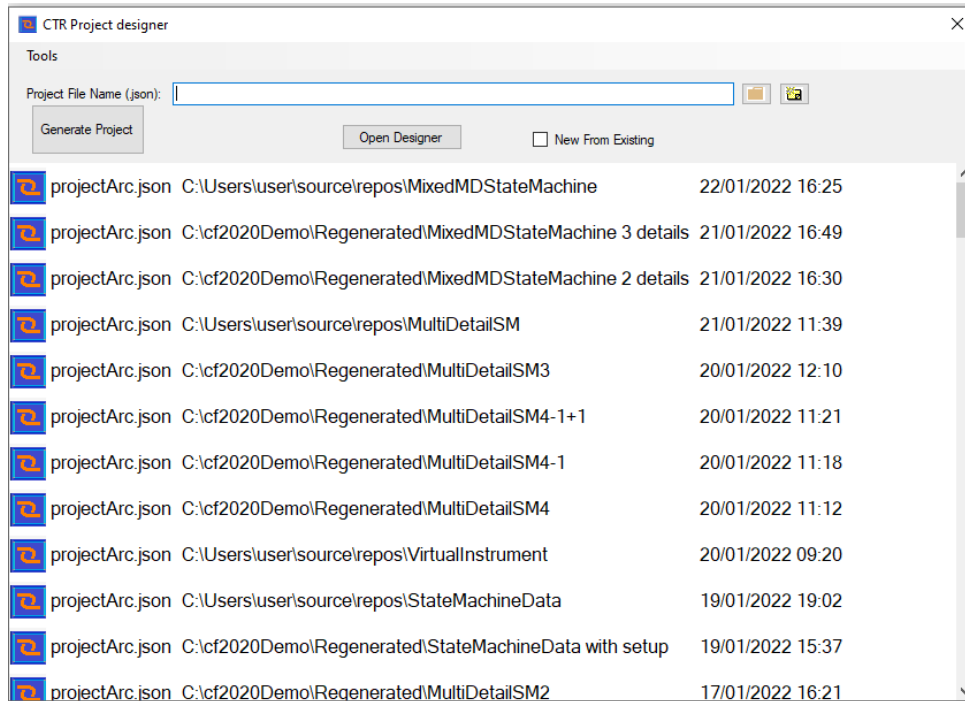


Figure 5.1: Example of the Dialog Display

When the directory is chosen the text box labeled Project File Name (JSON) is filled with the name of the directory combined with the name of the JSON Specification File that is always ProjectArc.json.

For each application which will be shown in the next sections, when the author completes the configuration of the wizard, the button "Save Project Space" must be pressed to save the project in the JSON Spec. File. Then the author can generate the code and build the canvas application.

Therefore, some examples of the applications built by the author, using the wizard and different models, are shown.

5.1.1. Parallel Command Template Manager

The Parallel Command Template Manager is a wizard that builds an application that allows executing tasks simultaneously on items that the user can select from groups displayed in the Main Selection Form when it is composed of a data grid or from a file when the Main Selection Form contains a list of directories in a tree-like view. The configuration of the wizard is shown in Fig.5.2.

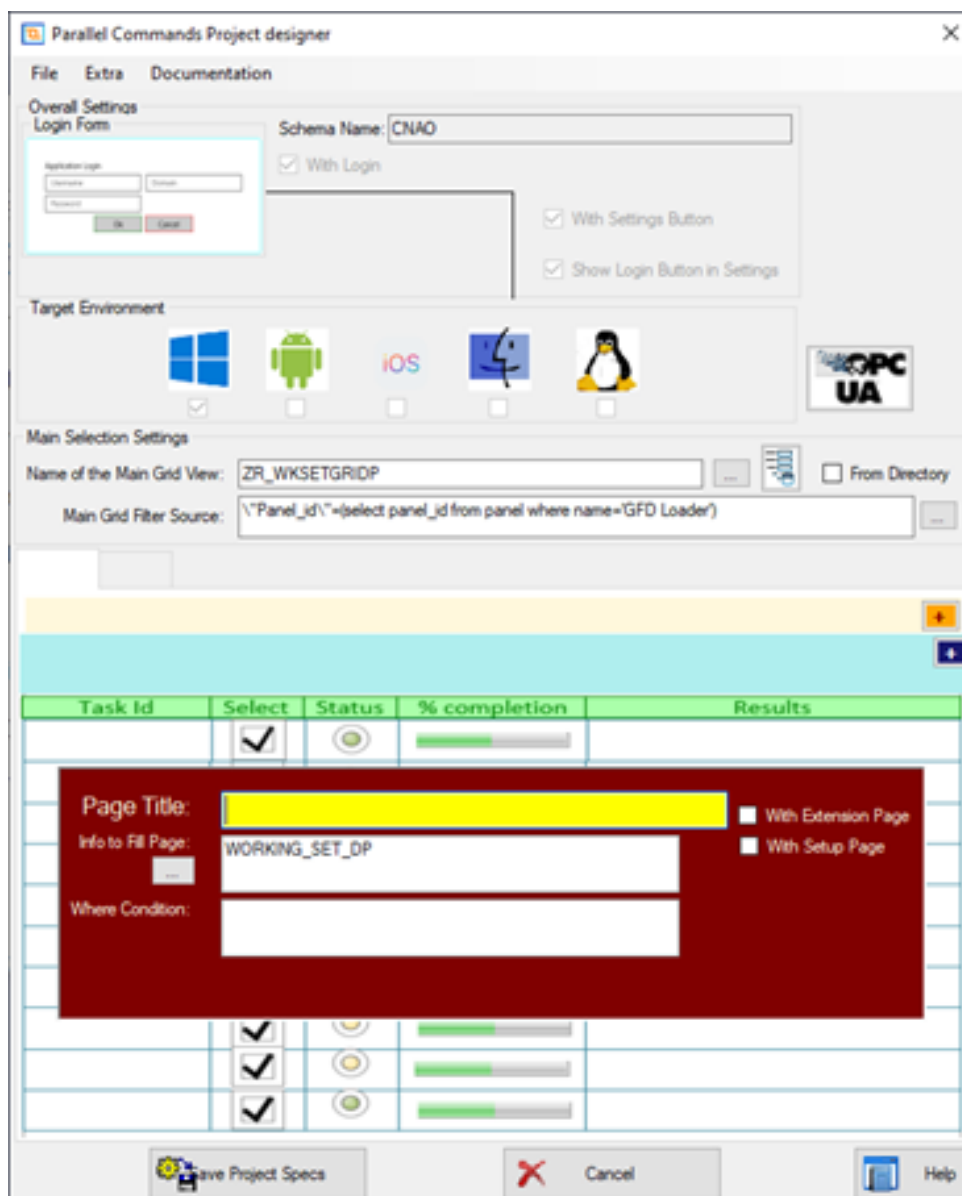
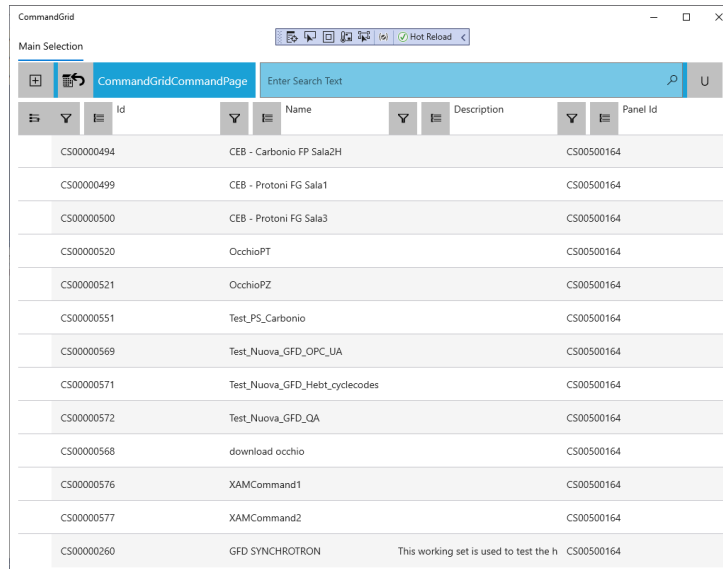


Figure 5.2: Parallel Command Project Designer

The application starts with asking the credentials. If the credentials are successful, a list of Working Sets or a list of excel files is shown to the user, as shown in Fig.5.3 and Fig.5.4.



Id	Name	Description	Panel Id
CS00000494	CEB - Carbonio FP Sala2H		CS00500164
CS00000499	CEB - Protoni FG Sala1		CS00500164
CS00000500	CEB - Protoni FG Sala3		CS00500164
CS00000520	OcchioPT		CS00500164
CS00000521	OcchioPZ		CS00500164
CS00000551	Test_LP_Carbonio		CS00500164
CS00000569	Test_Nuova_GFD_OPC_LUA		CS00500164
CS00000571	Test_Nuova_GFD_Hebt_cyclecodes		CS00500164
CS00000572	Test_Nuova_GFD_QA		CS00500164
CS00000568	download occhio		CS00500164
CS00000576	XAMCommand1		CS00500164
CS00000577	XAMCommand2		CS00500164
CS00000260	GFD SYNCHROTRON	This working set is used to test the h	CS00500164

Figure 5.3: Command Grid

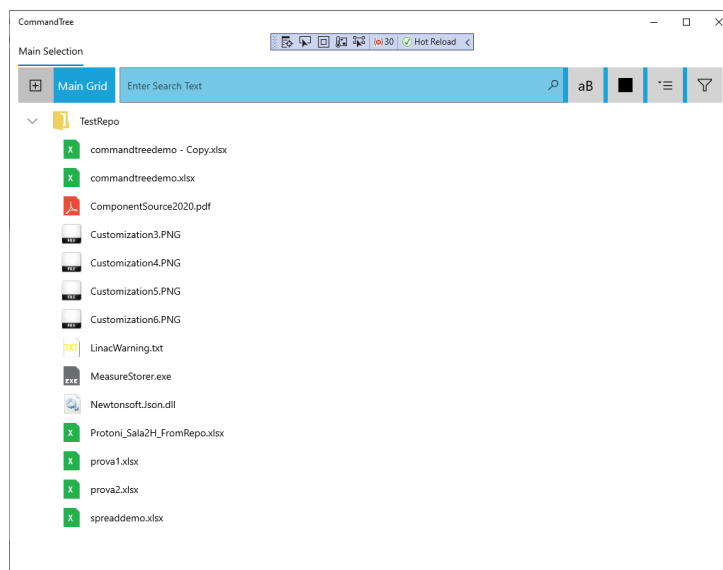


Figure 5.4: Command Tree

The Working Set is the identifier of a list of tasks or devices, the definition of which is generally contained in a database.

When the user has selected the desired item in any of the page types, the pages shown in Fig.5.5 and Fig.5.6 are displayed. In the page shown in Fig.5.5, each task is shown in a row with its initial status and all other columns empty.

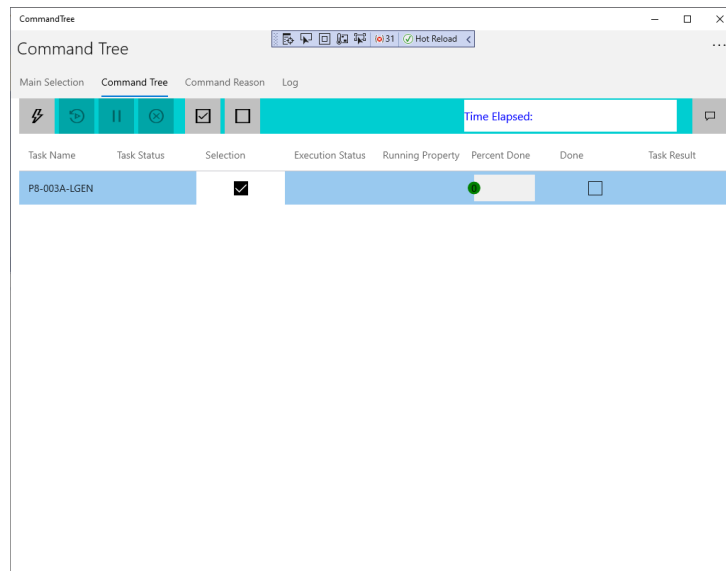


Figure 5.5: Task Status

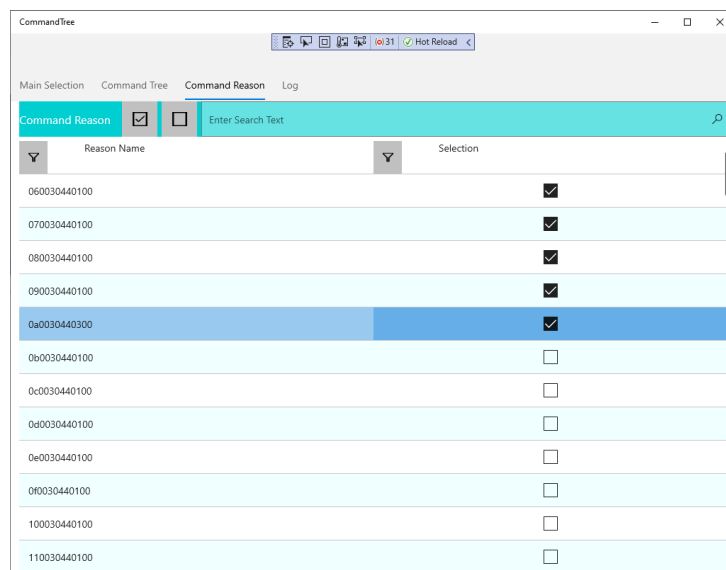


Figure 5.6: Task Utilities

The user then selects the task that shall be run by ticking the corresponding Selection column in Fig.5.5 and all the reasons for which each selected task must be run, in Fig.5.6. The pages shown in Fig.5.5 and Fig.5.6 can be accessed by clicking the corresponding page titles always present on top of the pages.

The tasks are run in parallel and the evolution is shown on the page by changing:

- The Execution Status that can be 'running' or 'done'.

- The Running Property that shows the reason for which the task is presently running.
- The Percent Done that gives an overview of the amount of the executed work.
- The Task Result that shows Success or Error depending on the outcome of the task.

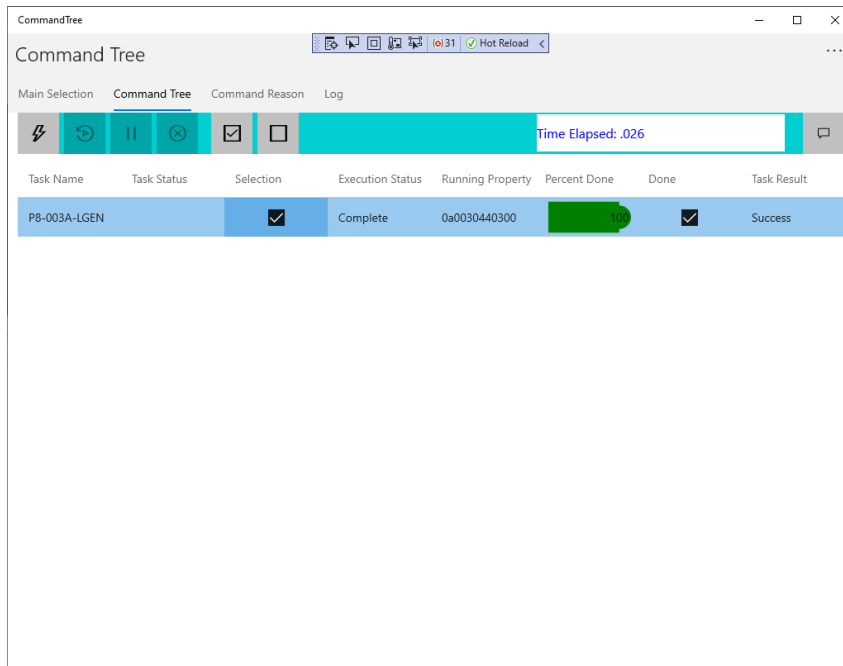


Figure 5.7: End of Execution

When the execution ends, the page can provide the information shown in Fig.5.7. The page shows the total elapsed time to complete all the tasks.

5.1.2. Data Uploader Template Manager

The Data Uploader Template Manager is a wizard that builds an application that allows uploading items contained in a file into a repository or downloading items from a repository into a file. The items are contained in a file the user can select in the Main Selection Form: the form contains a list of directories in a tree-like view.

The application is thus composed of a Login page if the developer has chosen to add it, then the Main Selection Form. When the user has made the choice, two other pages are built:

- A Spread page in which the items that are the target of the uploading or downloading activity are listed. The file, selected by the user in the Main Selection Form, that feeds the page both for uploading and downloading. When data are read from the

file, they are compared with the data contained in the repository and if they are equal the corresponding cell background becomes green; if the datum exists in the repository but the value differs the cell background becomes yellow; if the datum does not exist in the repository the cell background becomes red.

- A Log page that lists all the information that the application issues during the execution of the tasks.

The configurations of the wizard is displayed in Fig.5.8.

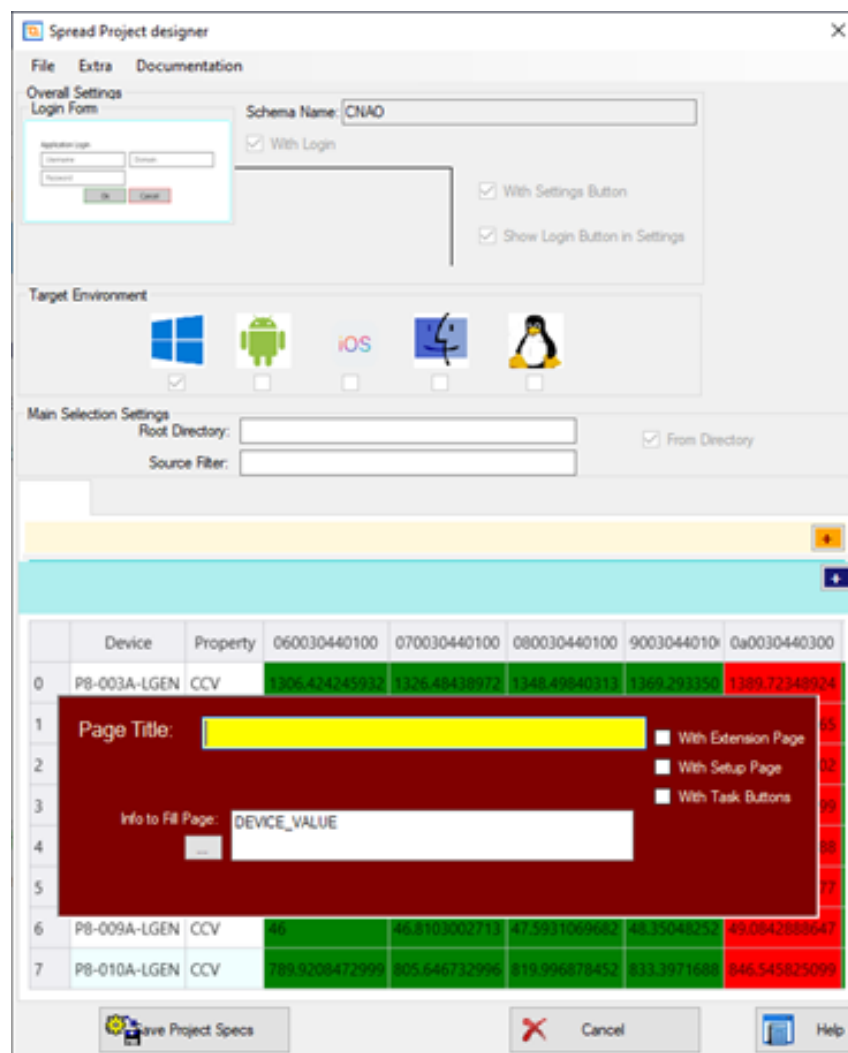


Figure 5.8: Spread Project Designer

The application starts showing the login page. If the credentials are accepted the page shown in Fig.5.9 is displayed.

The screenshot shows a web application titled "Spread Demo". It features a control bar with a lightning bolt icon, a play/pause icon, a stop icon, and a search icon. An "Elapsed Time" field is visible. Below the control bar is a table with the following data:

Device	Property	060030440100	070030440100	080030440100	090030440100	0a0030440100
P8-003A-LGEN	CCV	1306.424245932590	1326.464389720210	1348.498403130740	1369.293350135720	1389.7234892
P8-004A-LGEN	CCV	21	21.369919689081	21.72728796375740	22.073046372196	22.408044918
P8-005A-LGEN	CCV	53	53.93360683434720	54.835536289483	55.70816465363750	56.553637170
P8-006A-LGEN	CCV	29	29.51084147539750	30.00435004518880	32	30.944442979
P8-007A-LGEN	CCV	32.64	45	33.77041329224010	34.30782064707030	34.828504098
P8-008A-LGEN	CCV	51	60.74329377670770	61.75910134224750	64	63.694130587
P8-009A-LGEN	CCV	46	46.81030027132020	47.59310696823050	48.35048252957220	49.084288864
P8-010A-LGEN	CCV	789.920847299977	805.646732996447	819.996878452988	833.397168882852	846.54582509

Figure 5.9: Data Spread Application

5.1.3. List Monitor Template Manager

The List Monitor Template Manager is a wizard that builds an application that monitors the execution of tasks that can go through a fixed number of states: new, ready, done. The tasks are listed in a data grid and are shown when their state becomes new and are deleted from the data grid when their state becomes done.

The application is composed of a single page that can contain:

- the data grid,
- a header,
- a footer,
- a side pane on the left or on the right of the data grid that can contain an image and a multi-line text box.

The configuration displayed by the wizard is shown in Fig.5.10.

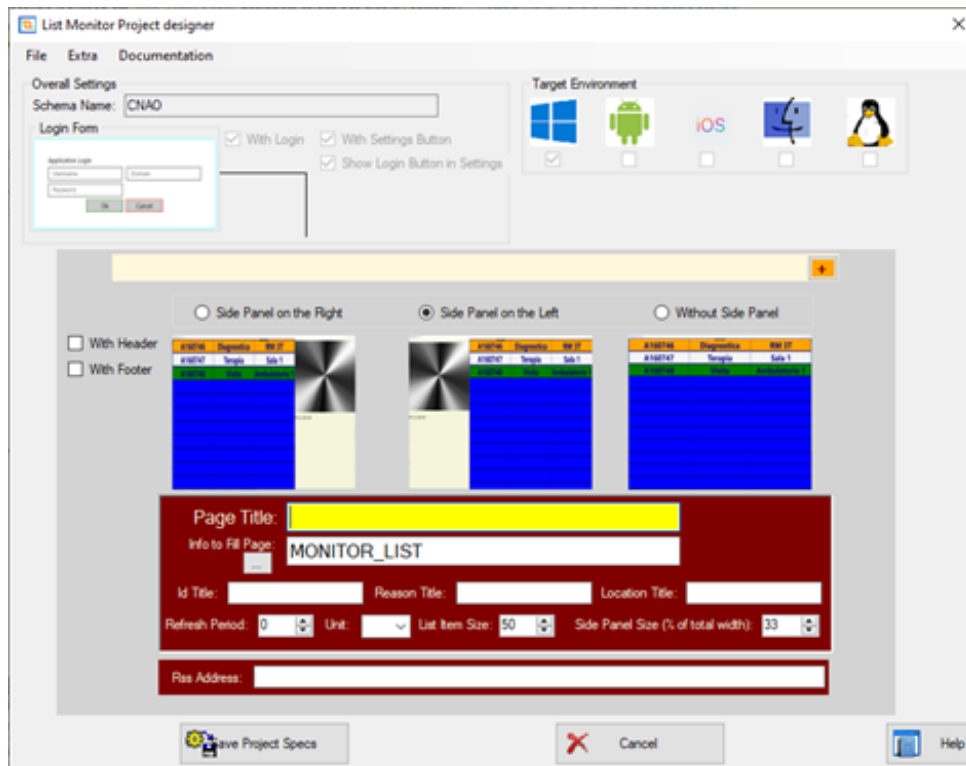


Figure 5.10: List Monitor Project Designer

The application is mainly a monitor, as shown in Fig.5.11; thus, no interaction is intended for an operator. It is still possible to add menus and menu items, mainly for configuration and maintenance purposes.

Patient ID	Reason	Room
A100002	Diagnostica	CT
A100000	Terapia	Sala 3
A100001	Visita	Ambulatorio 4

Covid, news. Fiaso: "Giù ricoveri del 17%. Calo più netto al Nord con -29%". LIVE

La curva dei ricoveri comincia a scendere rapidamente: in una

violenti" - Esplosione in un deposito a Pordenone: morta una donna - Migliori giovani università del m...

Figure 5.11: Monitor Application

5.1.4. Schedule Monitor Template Manager

The Schedule Monitor Template Manager is a wizard that builds an application that monitors the execution of tasks that are planned to be run during a day in a set of resources. Each task can be categorized into different types.

The tasks are listed in an agenda-like grid at the time they are planned and have a color that depends on their state ("to do", "done").

Moreover, there is an agenda for each resource and the agendas are positioned in parallel vertically on the screen.

The application is composed of a single page that can contain:

- the agendas,
- a header,
- a footer,
- a side pane on the left or on the right of the agendas that can contain a calendar and a multi-line text box that summarized the numbers of tasks done or the number of tasks to do grouped by type.

The Schedule Designer is shown in Fig.5.12.

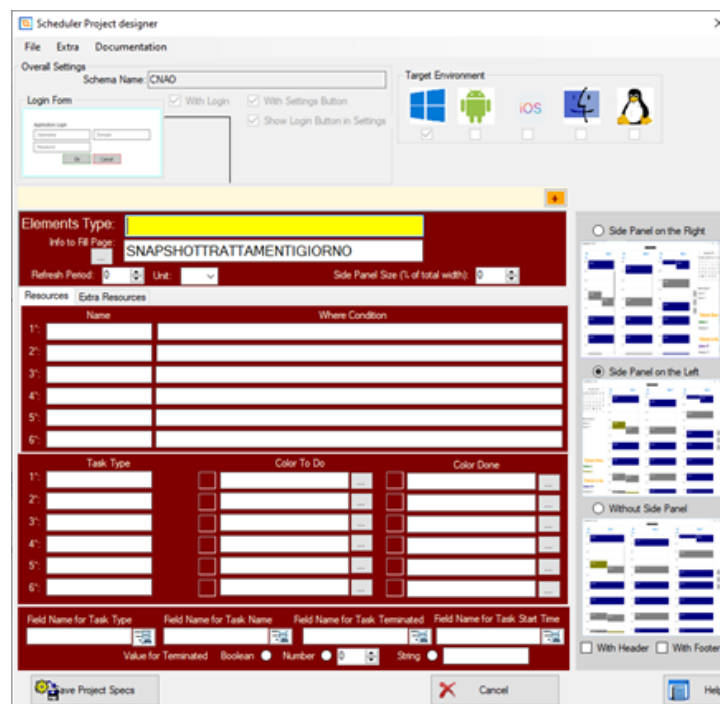


Figure 5.12: Scheduler Project Designer

The user can move the time table up and down using the arrows placed on the right side of the page. Moreover, the user can move the time table at the beginning of the working day and browse the calendar. The application refreshes the data in the page according to the parameters assigned by the developer.

The image of the application is shown in Fig.5.13

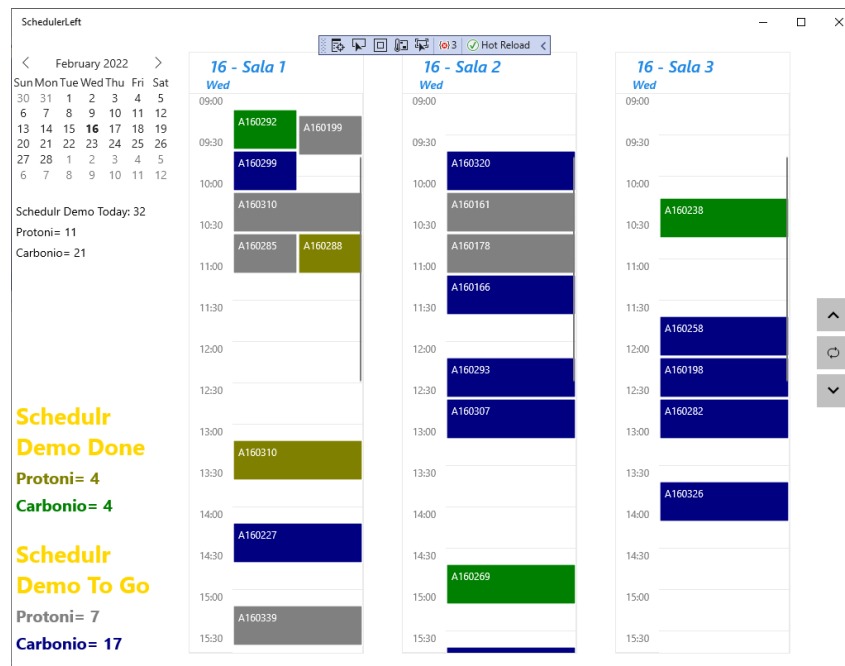


Figure 5.13: Schedule Monitor

5.1.5. Device Monitor Template Manager

The Device Monitor Template Manager is a wizard that builds an application that shows and monitors values of resource properties using a set of panels in which graphs, tabular views, or a panel in which different kinds of "View" instances can be bound to the values, can be displayed.

The developer can choose the layout and after that, the type of display must be assigned to each panel. The panels type can be grouped into three categories:

- graphs,
- data grids,
- data forms.

Graph panels can be of the following types:

- General Graph: contains plots, which are lines, in which both axes are floating-point values.
- Trend Graph: contains plots, shaped as lines, in which the horizontal axis contains values of time, the vertical axis contains floating-point values.
- Trend Bar Chart: contains plots, which are bar charts, in which the horizontal axis contains values of time, the vertical axis contains floating-point values.
- Category Bar Chart: contains plots, which are bar charts, in which the horizontal axis contains a set of discrete values (represented by strings), the vertical axis contains floating-point values.
- Category Chart: contains plots, which are lines, in which the horizontal axis contains a set of discrete values (represented by strings) and the vertical axis contains floating-point values.
- Values Bar Chart: contains plots, which are bar charts, in which both axes are floating-point values.

An example of the editing dialog layout is shown in Fig.5.14.

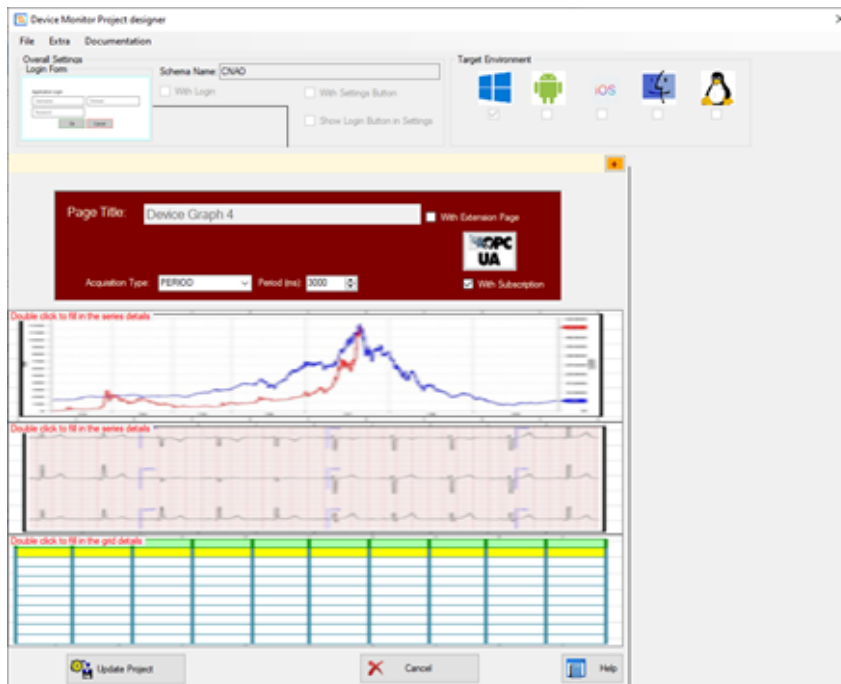


Figure 5.14: Editing Dialog Layout

The developer can assign:

- a title,
- the limits of the vertical axis,
- the resources (i.e. devices) and the properties of the value of which must be plotted,
- the color of the plotted values.

The user can interact, executing several actions on the graphs and the plots in each graph.

The graphs are refreshed according to the period assigned by the developer with the wizard. The types of graphs, grids and custom views depend on the choice in the wizard, as shown in Fig.5.15.

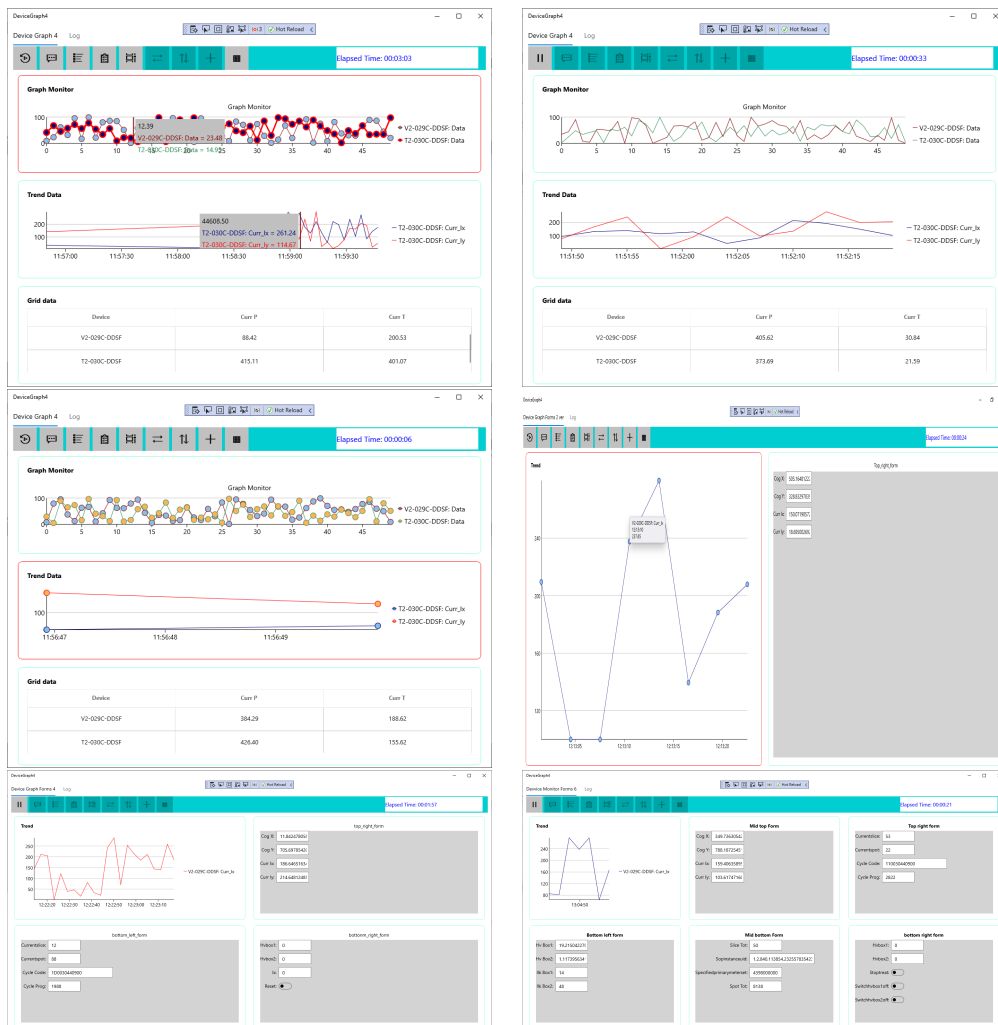


Figure 5.15: Different types of graphs

For each graph, a dialog box can be displayed showing a data grid containing the values of the series instances in the chart (see Fig.5.16). The color of the columns reflects the color of the series instances in the chart.

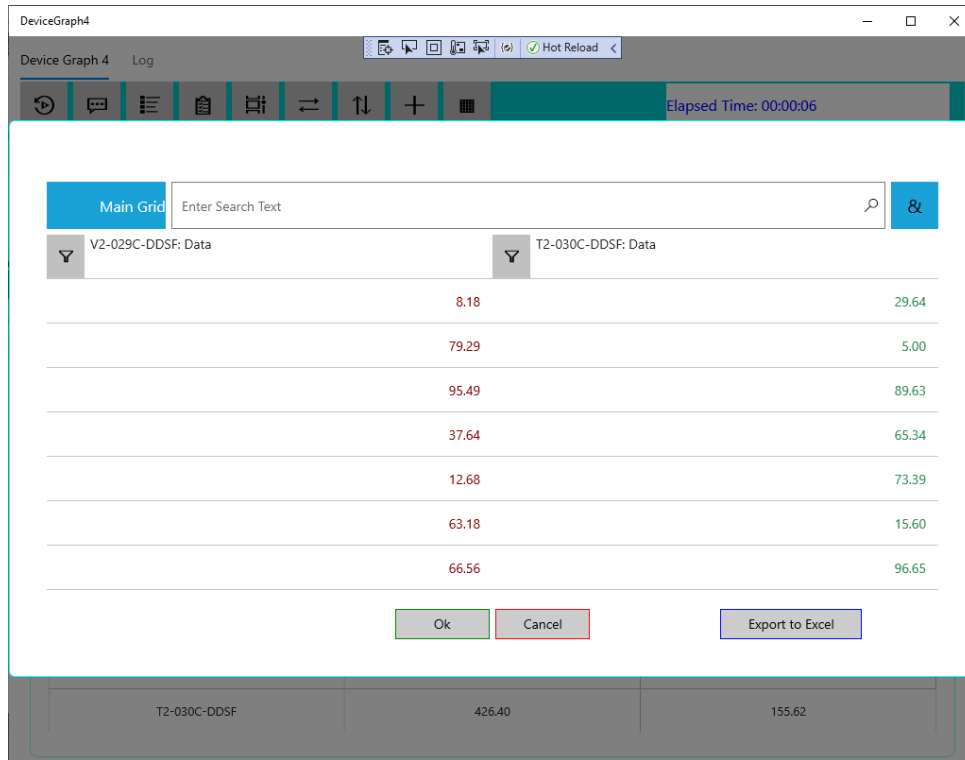


Figure 5.16: Series instances chart values

5.1.6. State Machine Template Manager

The State Machine Template Manager wizard allows building applications that manage a State Machine. The State Machine supported by the wizard allows defining:

- the name of the start state,
- the name of the transition,
- the name of the action managing the transition,
- the name of the state at the end of the transition.

The number of state/transition entries is arbitrary. The user interface of the State Machine can be composed of a single page on which all the "View" instances are deployed or can be split into two areas: the upper part in which the "View" instances are deployed as in the previous layout and a lower part that is occupied by a tabbed area in which the user can define an arbitrary number of panels that are managed by different "ViewModel"

instances. This way, the application can perform extra tasks while the State Machine is running.

The panels can be of different types. They can contain:

- a data grid,
- "View" instances, the content of which can be modified by the user,
- "View" instances to display data in different formats,
- "View" instances that are managed by the same "ViewModel" as the upper part of the page.¹

The panels can also contain data coming from a database and support for managing data edition is available when the data are shown in a data grid.

Extension forms can be added to the upper page and each pane separately. Menus and Toolbar buttons for executing extra commands can be added to the application. A Setup page can be added to configure the State Machine behavior.

A set of OPC-UA server classes can be added so that the corresponding definition files are automatically included in the project and the developer can interact with the OPC-UA server variables. Optional buttons are available for starting, suspending, resuming, and stopping the State Machine. They are contained in a toolbar at the top of the page that can be customized when defining the Toolbar buttons.

The State Machine can be run using four mechanisms:

- A configurable period after which the state of the machine is computed and if a new state is set, then the action related to the pair state/new state is fired.
- An event linked to a transition produced in the application (e.g., the change of a state value). The action corresponding to the pair state/transition is fired and the new state is set.
- An external event that is interpreted as a transition and the action corresponding to the pair state/transition is fired and the new state is set.
- The change in the value of a datum coming from an OPC-UA server is interpreted as a transition and the action corresponding to the pair state/transition is fired and the new state is set.

¹The "ViewModel" is the part of the application that creates the link between the "View" instances and the data coming from the field or the data computed by the business logic.

If the application interface contains "View" instances which enable the user to change data connected to OPC-UA Servers, two extra buttons are available:

- A Set All button to confirm the modified values and send them to the target destination.
- A Clear button to cancel the modified values and show the original values again.

The model supporting the production of State Machine applications without the lower tabbed page is called State Machine and its interface is shown in Fig.5.17. The model supporting the production of State Machine applications with the lower tabbed page is called Multi-Detail State Machine and its interface is shown in Fig.5.18.

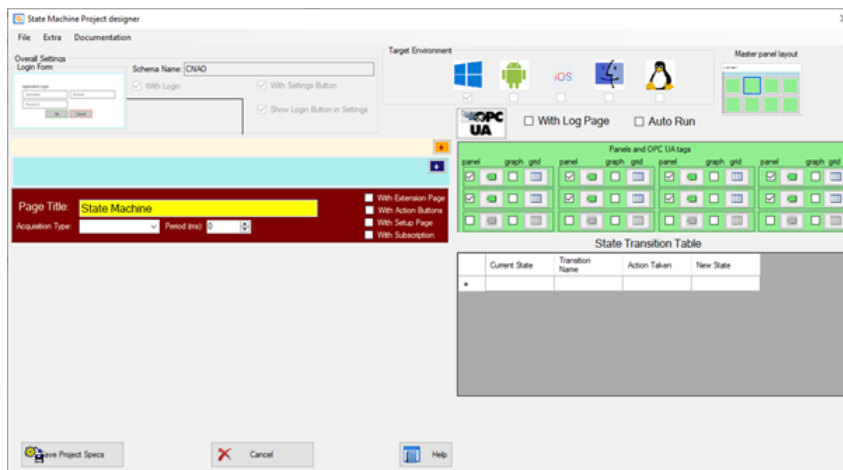


Figure 5.17: State Machine Layout

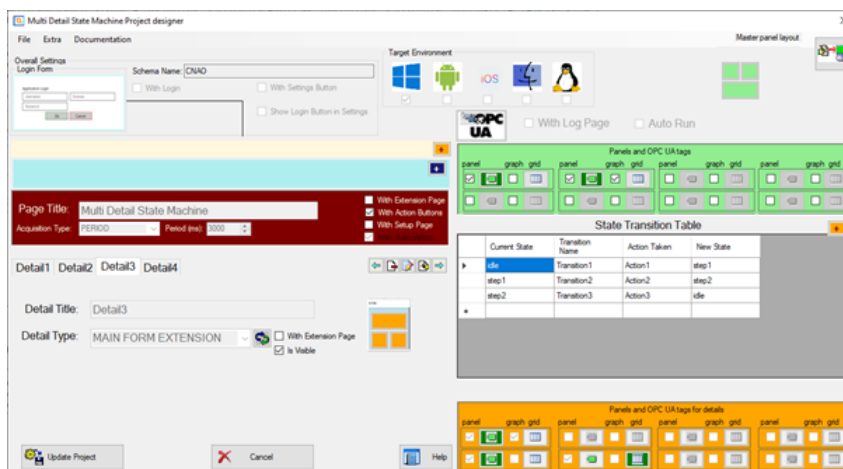


Figure 5.18: Multi-Detail State Machine Layout

The application shows an interface like the one displayed in Fig.5.19 in the case of State Machine type. The State Machine can be started, suspended, resumed, or aborted using the buttons present in the upper toolbar.

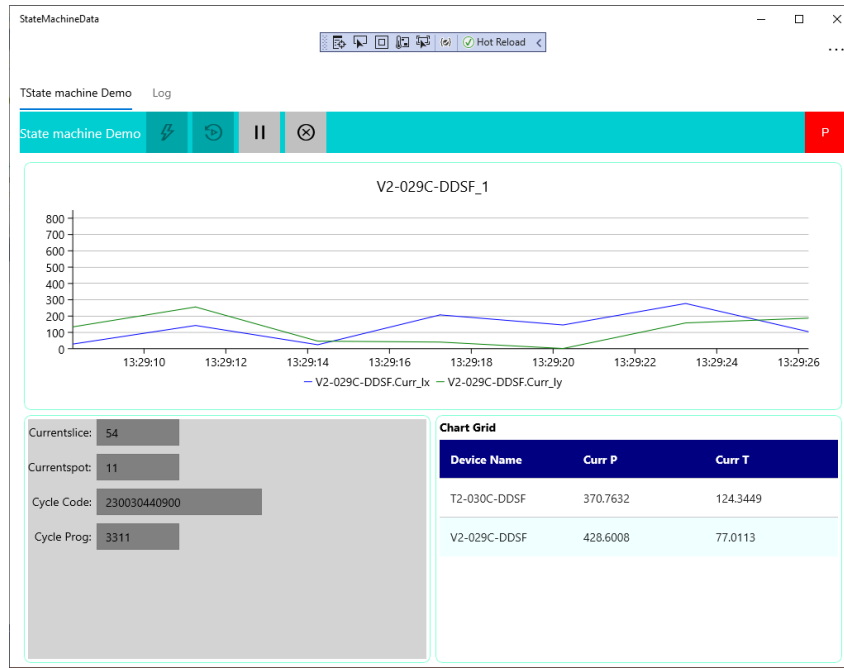


Figure 5.19: State Machine Data

The application shows an interface like the one displayed in Fig.5.20 in the case of Multi-Detail State Machine. The behavior of the action buttons, if present is the same as for the case of the State Machine.

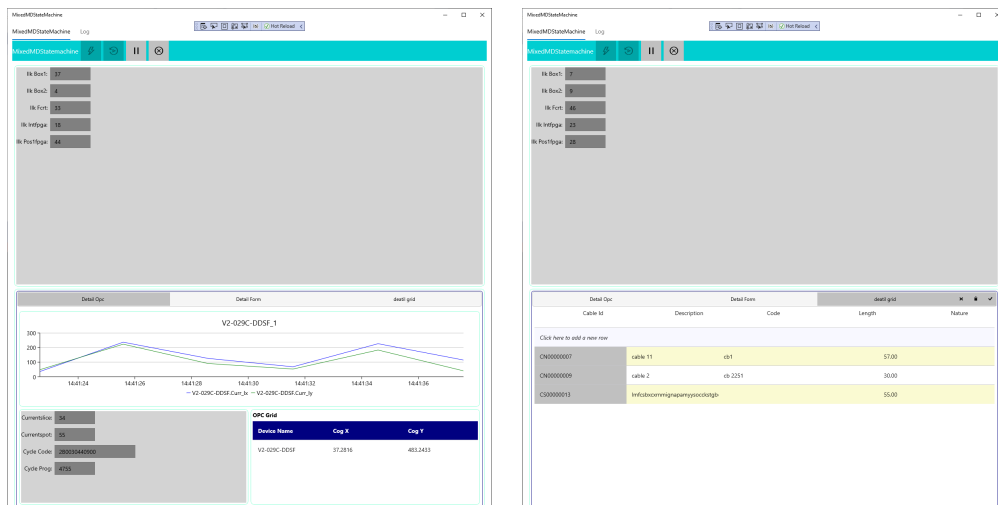


Figure 5.20: Multi-Detail State Machine Data

5.1.7. Procedure Template Manager

The Procedure Template Manager wizard allows building applications that manage the execution of activities on a set of devices.

The activities are grouped into steps. Each step has a working set associated. A working set is a group of devices on which the activities of the step are performed. The same steps can be applied to a set of lines. In each line, the working sets associated with the steps can vary.

The wizard can define an arbitrary number of lines and an arbitrary number of steps per single line. Therefore, the activities depend solely on the application.

The layout of the Procedure Template Manager wizard is shown in Fig.5.21.

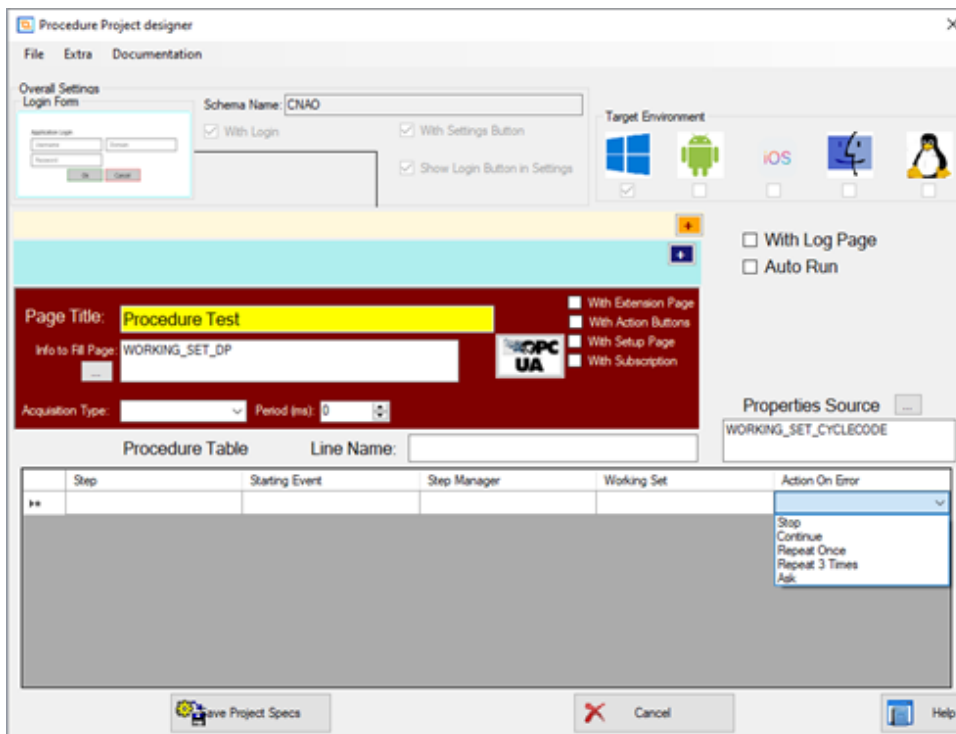


Figure 5.21: Procedure Project Designer

The user must choose the line for which the procedure will run. At the end of each step, depending on the choice of the developer the further step can be started automatically or a request can be shown to the user, as shown in Fig.5.22 and Fig.5.23.

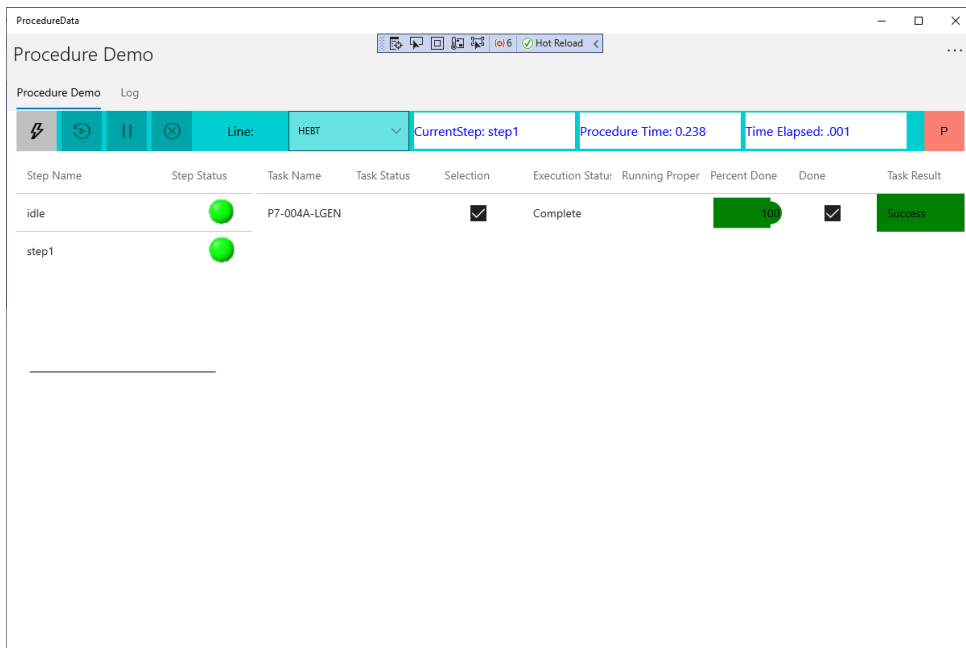


Figure 5.22: First step example

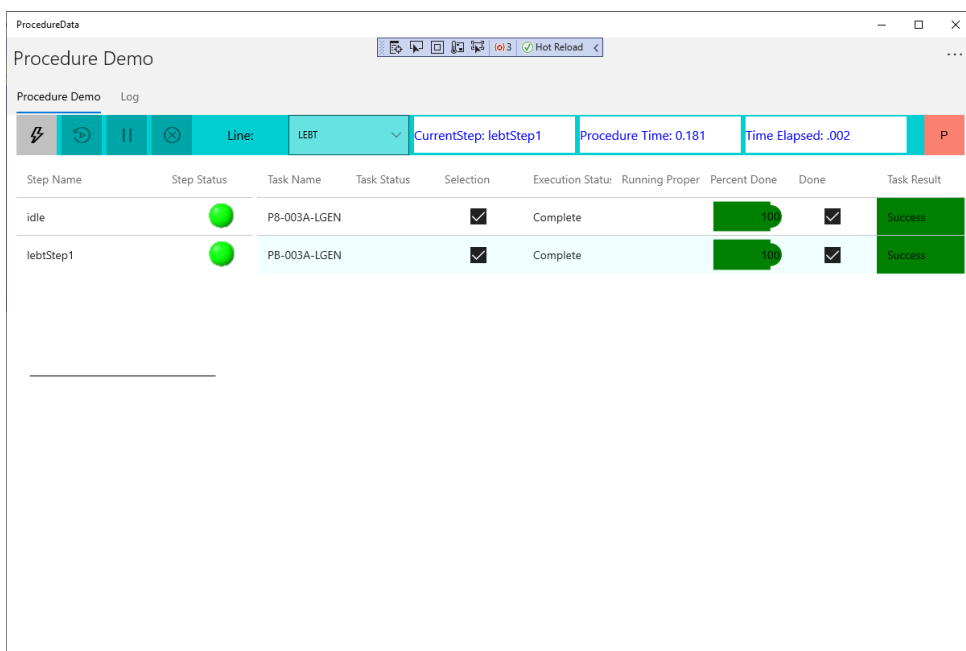


Figure 5.23: Second step example

In the case of error during one step, the behavior of the application depends on the choice of the developer. The procedure can stop or a message can be shown to the user asking if the procedure must continue or the error can be logged, still continuing the execution.

5.2. Requirements and software release

At the end of the process of drafting the JSON Specification File, all the tests, the codes relating to the applications and the documentation are taken and conveyed to the Jira software, already managed by the CNAO foundation to set the requirements and carry out the necessary checks.

The main purpose of the software requirements is to provide a rational explanation and a functional description without defining in depth the characteristics of the SW. Therefore, the list of software requirements must explain all the functions of the software indicating and describing all the requirements of the device without going into the merits implementation of the SW.² An example of the list of the requirements of the Jira software is shown in Fig.5.24.[51]

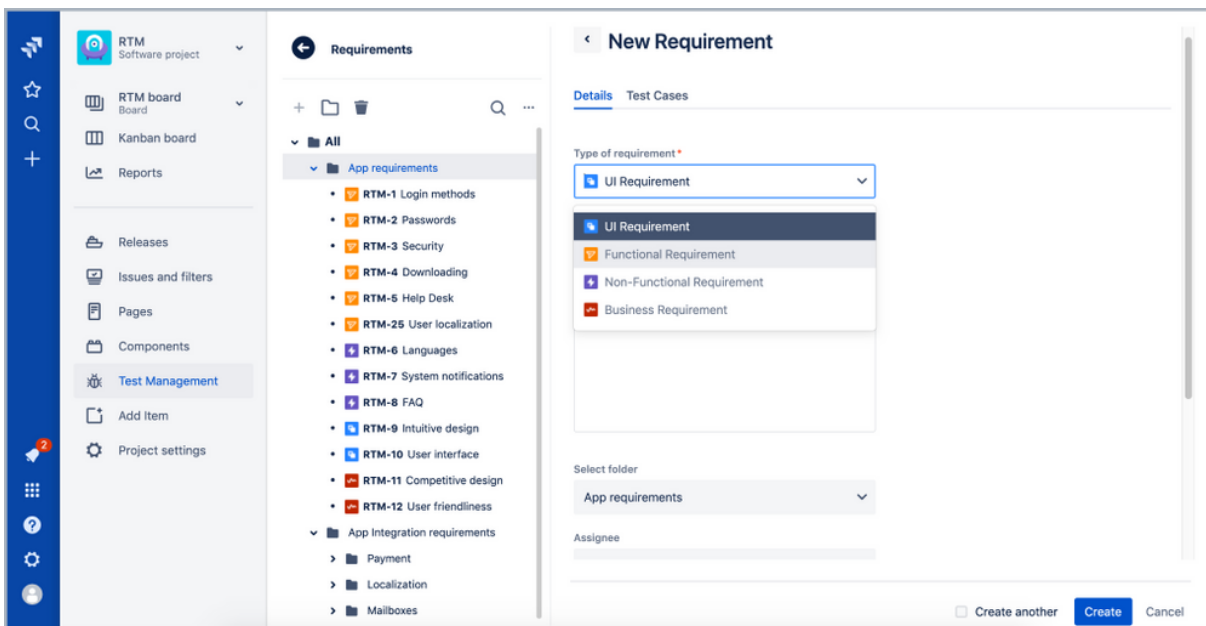


Figure 5.24: Jira Environment & Requirements

After having carried out the necessary checks, the developer must complete a document of the missing parts and answer the self-explanatory questions that constitute a check-list for the software release.

²It is useful to underline that this part of the validation of the software documentation is not the objective of the work done in the thesis.

5.3. Results & Final Remarks

The applications, created by the author, have already been tested by the CNAO Center and have already been simulated by the control group present within the Institute.

Therefore, the applications are fully functional and are consistent with the process map, concerning the Product Line Architecture and the *Agile* development approach, described by the author in Sec.4.4 included in Chap.4.

6 | *Agile* implementation in the Aeronautics and Space field

In this chapter a project modeling methodology will be analysed in the aeronautical and space field, in order to be able to implement and discuss the *Agile* philosophy in these types of industry. The author of the thesis is the responsible for highlighting the similarity of the approaches used by the toolkit for accelerator control systems and by MBSE¹ for avionics.

6.1. Systems engineering

In the technological industries, a project aims to create a unique product or service that can be seen as a system. A system is a set of elements, including people, hardware, software, structures, procedures and documents, which together produce results which cannot be obtained separately from these elements. They are structured and linked together in order to allow the fulfillment of a function necessary to satisfy the needs of customers, which define the requirements of the project. The result achieved by this set of elements has a greater value than the sum of the values of the individual parts, because by combining them together an added value is obtained, thanks to the interrelationships that arise from the connection of the elements.

In aeronautics and space, these systems have a function related to the transport of people, supply of energy, transmission of messages and more. To obtain these results, the system reaches considerable dimensions and requires the use of a very high number of elements and therefore a number of relationships of the order of millions, increasing the level of complexity of the system. The work team, including engineers, scientists and project managers, finds itself facing new challenges in which they are required to have, in addition to technological knowledge, a leading role in order to be able to independently take the

¹Model-Based Systems Engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.

initiative to work and guide others in the complexity of these systems.

In the space field, an example of a system is the re-entry vehicle used to re-enter the Earth from space.

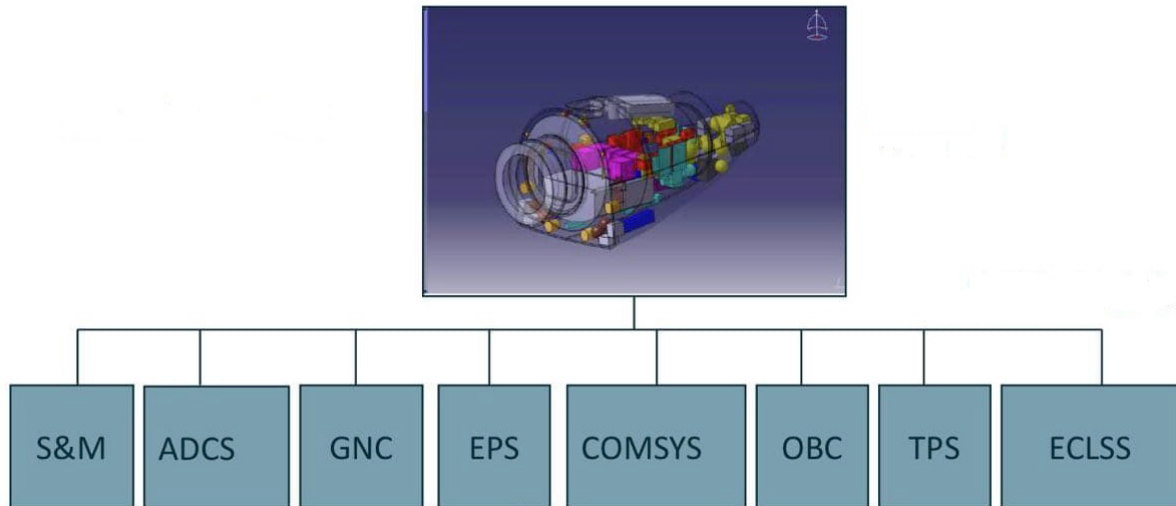


Figure 6.1: System of a vehicle used for re-entry from the atmosphere

Fig.6.1 shows the subsystems that the vehicle contains. The subsystems reported are:

- Structures and Mechanisms System (S&M);
- Attitude and Determination & Control System (ADCS);
- Guidance, Navigation and Control System (GNC);
- Electric Power System (EPS);
- Communication System (COMSYS);
- On Board Computer (OBC);
- Thermal Protection System (TPS);
- Environmental Control and Life Support System (ECLSS).

In this case the elements of the system are its subsystems, but in turn these are a system made up of subsystems. This decomposition can be carried out down to the final element itself and this gives an idea of the degree of complexity of an aerospace system.

6.2. The importance of the V-Model

To visually describe the lifecycle of a system's development, the V-model can be used, which can be adapted to any project. The V-Model is shown in Fig.6.2

This representation describes the activities to be performed and the results to obtain during the development of the product or service.

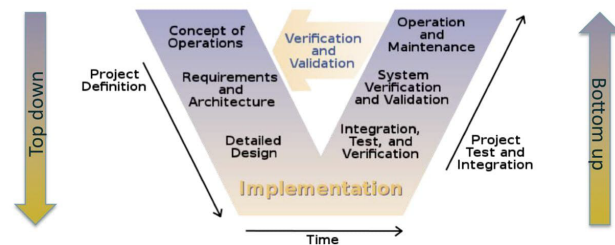


Figure 6.2: V-Model scheme

The left part of the model (Project Definition) represents the initiation and planning phase, in which the requirements are written and the system is defined also through the use of computer models and the next section reveals how to carry out this work through the use of the MBSE. The basis of the model (Implementation) represents the phase of execution, implementation of the system and then begins to produce. The right part of the model (Project Test and Integration) represents the integration phase of the parts in which the validation of the components is carried out through specific controls and finally the system built for maintenance purposes is monitored.

6.3. Model-Based Systems Engineering

The implementation of the *Agile* Project Management involves changing the focus of the main project work from the design itself to the development of the final product, from a Project-Based to a Product-Based philosophy.

Model-Based Systems Engineering functions perfectly thanks to its characteristic of being a holistic engineering system that focuses entirely on the modeling of the system and its development. Moreover, it uses virtual models to develop the architecture of a system and for the implementation of the activities on the prototype that take place during the lifecycle of the project which are focused on the analysis of requirements, trade-offs, study of the design, verification and validation. In short, a tool capable of optimizing the speed, costs and quality of project development thanks to its revolutionary and effective features.

In a MBSE system the model is the source of all information useful for the development of the system. This incorporates different perspectives of study of the system that are complementary and necessary to better respond to the needs of the stakeholders. No more different models are developed but only one that evolves according to the needs of

the stakeholders. Consequently, all the information is concentrated in a central repository which leads to different capacities.

This working methodology increases the identification of all aspects of the system to 90% already at the first analysis, thus allowing to build models with a halved number of problems compared to one whose writing of the requirements was done through a traditional approach.[52]

6.3.1. Main features and motivation

The virtual model of the system being studied is generated at the beginning of the development phase and can be considered as a delegated substitute of the real system to carry out a design study, analysis, validation and user training, which grows and evolves with the increase of the characteristics of the project that are defined during the life cycle of the project.

To define an MBSE model, there are different challenges to be faced. Therefore, the definition of the model and its purpose take place on the basis of a set of questions created specifically to respond to the requests of the stakeholders, which translate into project requirements.

It is said that the model is appropriate when it is able to answer all the questions forming part of the set in a satisfactory way, and that it is consistent when all the possible ambiguities that could arise, for reasons of language or other, are clarified.

The model of the system, which most of the time is abstract, becomes more and more concrete as the development process progresses. Moreover, it must describe and represent the information relating to the system and its interactions with the surrounding environment based on the views of the various stakeholders. This feature allows the model to be flexible and highlight only the information that is useful to the point of view of the stakeholder who is studying it, leaving useless information further behind, in order to carry out its function correctly.

The main reasons for the use of MBSE derive from the need to identify all the possible problems in terms of costs and timing that can in some way also damage the design and architecture of the system model and therefore the project itself.

The Model-Based methodology is excellent for solving these problems thanks to several functions. First, from the architectural point of view, this approach facilitates the representation of a system thanks to the possibility of reusing models already defined as representative models of the components of the architecture and also thanks to the ability

to grow together with the complexity of the model, starting from architectures with simple components, moving onto its subsystems and finally to the realization of the components.

Secondly, these templates provide visual information, such as the inter-relationships between components and scale views of the model, which are subsequently translated into useful data to describe the model through assumptions, requirements, design and analysis.

Thirdly, from the point of view of documentation, this methodology allows the automatic generation of documents in order to inform stakeholders about the development of the model. These are documents which, thanks to the characteristic of the interconnections between the parts of the model, allow to keep track of the development of the project in a complete way, taking into account all the information provided, the data generated and the decisions made.

Finally, the use of models with multiple interfaces can facilitate collaboration between individuals with experience in different disciplines since the projects in the engineering field are the result of the union of several scientific fields and the collaboration between these is significant for the success of the project. The MBSE makes it possible to achieve effective communication between all participating parties, the individuals and organizations involved in all phases, from design to maintenance. This milestone is achieved thanks to the sharing of information common to the context related to the requirements, the business context and the mission, usage scenarios and performance measures.

6.3.2. Benefits implementation

The main objective of the MBSE is to be more efficient than the methodologies of traditional modeling languages, such as the Systems Modeling Language (SysML), working more accurately where these have shortcomings, such as inconsistencies between the models chosen for the representation of the different disciplines and inconsistent, outdated and disconnected documentation. Its purposes are:

- Facilitate comprehension and provide insights.
- Allow communication by explicitly stating all aspects of a design.
- Support visualization of evolving system design and related documentation.
- Allow the verification and validation of the requirements, structure and operation of the system
- Track behavior against requirements.
- Resolve discrepancies and support performance analyzes.

The work is based on a collection of integrated models collected in a repository and seen as the only source to follow. These models are left to evolve in order to answer a growing number of stakeholder questions.

MBSE provides several benefits and advantages over a model that is based on traditional systems, such as:[52]

- transparency and traceable design,
- possibility of permanently archiving standard and customized MBSE models in a specific repository,
- creation of views built specifically for the end-user,
- automatic management of the type of model configuration,
- support for different types of project and lifecycle methodologies,
- reduction of costs and time along with an increase in quality.

6.3.3. MBSE in the present projects

The MBSE is used in American and European companies in order to create a system architecture that can fully cover the requirements of the project. In the development of complicated large projects, the use of MBSE is limited to specific areas where it is necessary to define the requirements, the architecture of a system, the functional diagram, software and hardware behavior, data flows and activities of test. In some cases its use is also extended to work on mechanical structures, thermodynamic problems and other engineering subsystems.

The International Council on Systems Engineering (INCOSE) was one of the first organizations to create MBSE development teams. In 2013 the IEEE² Systems, Man and Cybernetic Society created the MBSE Technical Committee to assist in the development of the MBSE. In the aerospace field, the largest entrepreneurial companies, such as Lockheed Martin, Boeing and Northrop Grumman along with some universities, offer courses on learning the MBSE. Even in major research and development centers, such as NASA Jet Propulsion Laboratory and NASA Glenn Research Center, the use of the MBSE it is a daily practice.[53]

Its implementation in large projects is an important challenge to face because large dimensions mean a large number of interactions between the elements that make up the

²IEEE, which stands for Institute of Electrical and Electronic Engineers, is the most important organization in the world in the field of electrical and electronic engineering and information technologies.

system. The transition to the MBSE requires a cultural change during the development of the system, it is necessary to train the company team in the use of the MBSE in order to develop confidence in this working methodology. Other problems are due to the various changes that arise, especially those relating to the documentation that is produced following the evolution of the model with an MBSE system, but also to the use of unknown tools to describe the model. Especially in the case of large projects defined by a high number of elements, it is necessary to use innovative and non-linear modeling methodologies.

Complex systems tend to have a behavior defined as "emergent", a behavior that is not explicitly defined and is not even a direct consequence of its elements but a result of the interactions of the components of the system that make up the model with each other and with the external environment. This is not clearly defined because often the limits of the system vary or are unstable. These features pose a particularly high modeling challenge because one must take into account the dynamism of the systems that complicate the validation, testing and evaluation process.

An important challenge is introducing a descriptive and analytical model within the MBSE that is able to take into account the human factor, its capabilities and limitations within the models. One of the first companies that is working on integrating this feature is Airbus, developing the Human System Integration (HSI). The aim is to insert the human component into the architecture, that defines the system model using the MBSE language.

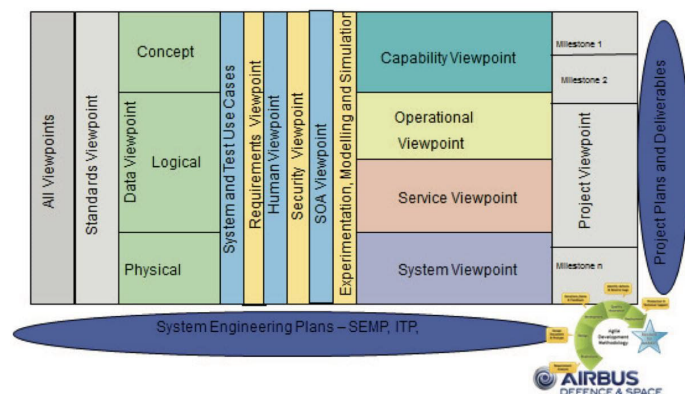


Figure 6.3: Framework defined by Airbus DS[54]

The Fig.6.3 shows the typical functional architecture of a system defined via MBSE by Airbus DS.

6.4. Final remarks

The transition from a working methodology based on the project itself and its documentation, defined Document-Based Systems Engineering (DBSE), to the MBSE methodology requires a high and constant commitment that a company is not always willing to face.

A more typical situation that occurs is one in which this change is applied only in the

areas where its effect is most notable, applying communication strategies and strategies of reuse of previous jobs in order to reduce the work and make it more effective.

In conclusion, leveraging a Model Based Systems Engineering solution, such as the toolkit for accelerator control systems described in the previous chapters, allows companies to reduce complexity, rework, and quality issues while meeting regulatory compliance and lowering risk.

7 | Conclusions and future developments

It was possible to see how the creation of a Specification File, through a wizard, greatly facilitates the validation system of a product. After that the author discussed how *Agile* development methods work in Chap.4 it is possible to conclude that this approach is very important to attempt to reduce the time and the effort in centers and in companies which face daily projects, which require software in *Mission Critical* environments.

Furthermore, the work, carried out by the author in the previous chapters, is preliminary to the final realization of the product. The subsequent development work will be completed by engineers, which will embrace this project in the future, at the CNAO Foundation, in order to guarantee the realization of the final product and the final validation of the software. Besides, the work presented in this thesis had the goal of developing tools, services, and components for the upgraded environment, while the total project encompasses the development of the environment's applications, as well as performing their respective medical software certification tasks.

In the near future, several applications of the legacy will have to be reviewed, and the total amount of new required applications will have to be analyzed by the control system stakeholders. During the development, metrics can be gathered, and the Product Line approach can be further evaluated. Particularly, metrics such as required number of hours for development and certification, as well as number and severity of errors encountered during testing would provide insight on the impact of the architecture and, therefore, in the creation of new wizards to improve the entire system.

Regarding the developments in the aerospace and space fields, the future aim of the author would be to adapt the technical sub-level generated by the JSON Specification File, containing all the documents for validating the product, the tests and the codes of the applications in further *Mission Critical* projects. This system could be used to investigate the possibility of its usage in some aerospace projects, in order to implement it in the *Agile* methodology, facilitating the work and shortening development times.

Bibliography

- [1] Carsten Welsch. *Optimization of medical accelerators*. A Marie Skłodowska-Curie European Training Network. OMA Brochure, Update 2019, pp. 9.
- [2] Alberto Degiovanni; Ugo Amaldi. History of hadron therapy accelerators. *Physica Medica* 31 (2015) pp. 322-332, 2015.
- [3] Provision Cares Cancer Network. Proton therapy advanced radiation therapy treatment. URL: <https://provisionhealthcare.com/about-proton-therapy/advantages-of-proton>.
- [4] Carlos Eduardo Fernandez Afonso. *Product line architecture for Hadrontherapy control system: applications development and certification*. PhD thesis, Università degli Studi di Pavia, Dipartimento di ingegneria industriale e dell'informazione, 2019.
- [5] A. J. Lennox; F.R. Hendrickson; D.A. Swenson; R.A. Winje; D.E. Younge. Conference: *Proton linac for hospital-based fast neutron therapy and radioisotope production*. Villigen (Switzerland), 1989. International heavy particle therapy workshop, Fermi National Accelerator Lab. (FNAL), Batavia, IL (United States). TM-1622, pp. 1-3.
- [6] D. Ungaro; A. Degiovanni; P. Stabile. Light: A linear accelerator for proton therapy. *North American Particle Accelerator Conference (NAPAC'16), Chicago, IL, USA*, pages 1282–1286, 2016,2017.
- [7] A. Degiovanni et Al. "A Cyclotron: LINAC complex for carbon ion therapy". *Workshop "Physics for Health in Europe" ID:65*, pp. 1-2, Wednesday, February 3, 2010 6:15 PM.
- [8] U. Amaldi et al. Accelerators for hadrontherapy: from lawrence cyclotrons to linacs. *Nucl. Instruments Methods. Phys. Res. Sect. A Accel. Spectrometers. Detect. Assoc. Equip.*, 620(2-3):563–577, 2010.
- [9] IBA. "IBA worldwide - Shaping the future of proton therapy.". URL: <https://ibaworldwide.com/proton-therapy>.

- [10] Varian. "Proton Therapy | Varian Medical Systems.". URL: <https://www.varian.com/oncology/solutions/proton-therapy>.
- [11] WIPO IP Portal. Energy modulation of a cyclotron beam. URL: <https://patentscope.wipo.int/search/en/detail.jsf?docId=W02019016249>.
- [12] Particle Therapy Co operative Group. Particle therapy facilities in clinical operation. URL: <https://www.ptcog.ch/index.php/facilities-in-operation>.
- [13] R. Muller. *Control Systems for Accelerators: Operational Tools*. pages 629–670, 2016.
- [14] ICALEPCS "The International Conference on Accelerator and Large Experimental Physics Control Systems". *ICALEPCS - About*. URL: <https://www.icalepcs.org/icalepcs.html>.
- [15] B. Kuiper. Issues in accelerator controls. *International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS-91)*, pp. 602-611, 1991.
- [16] L. R. Dalesio; M. E. Thuot; Los Alamos National Laboratory. Control system architecture: the standard and nonstandard models. pages 602–611, 1993.
- [17] L. R. Dalesio; A. J. Kozubal; and M. R. Kraimer. Epics architecture. Conference: *Proceedings of International conference on accelerator and large experimental physics control systems*. Reference Number: 25028669, pp. 1-6, 1991.
- [18] Tango Community. Overview of tango controls. URL: <https://tango-controls.readthedocs.io/en/latest/overview/overview.html>.
- [19] Wikipedia the free Enciclopedia. Common object request broker architecture. URL: https://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture.
- [20] Fondazione CNAO. *The History of CNAO*. URL: <https://fondazionecnao.it/en/history>.
- [21] IAEA International Nuclear Information System. *The EULIMA project*. URL: https://inis.iaea.org/search/search.aspx?orig_q=RN:19078207.
- [22] U. Amaldi. *History of hadrontherapy in the world and Italian developments*. *Riv. Medica*, vol. 14, no. 1, pp. 7-14, 2008.
- [23] U. Amaldi. *Gli acceleratori nella terapia dei tumori*. *Rivista di cultura e politica scientifica*, N. 2-3/2008, pp. 109-119, 2008.

- [24] Fondazione CNAO. *Descrizione del dispositivo Ministero della Salute*, pp. 3-28. Strada privata Campeggi 27100 Pavia.
- [25] Sunpower electronics. *What is Ripple?* URL: <https://www.sunpower-uk.com/glossary/what-is-ripple/>.
- [26] L. Casalegno; M. Pezzetta; and S. Toncelli. *CNAO General Control System Organization Document - Unpublished internal document*. 2003.
- [27] L. Casalegno; M. Pezzetta; and S. Toncelli. *Timing and Signal Distribution Services Requirements Specification Document - Unpublished internal document*. 2004.
- [28] Fondazione CNAO. CNAO General Control System organisation document, pp. 10-23. Technical Report 2, 8 2004.
- [29] C. Afonso; L. Casalegno; S. Foglio; S. Gioia; M. Necchi; C. Larizza. *Integrating mobile devices into CNAO's control system, a web service approach to device communication*, pp. 1-5, vol.2. ICALEPCS 2019.
- [30] NDV. "Medical Devices Regulation (EU) 2017/745 - MDR". URL: <https://www.dnv.it/services/medical-devices-regulation-eu-2017-745-mdr-138310>.
- [31] European Parliament and the council on medical devices. "*Regulation (EU) 2017/745 of the European parliament and of the council on medical devices, amending Directive 2001/83/EC, Regulation (EC) No. 178/2002 and Regulation (EC) No. 1223/2009 and repealing Council Directives 90/385/EEC and 93/42/EEC*", Annex V, pp. 114, 5 April 2017.
- [32] European Parliament and the council on medical devices. "*Regulation (EU) 2017/745 of the European parliament and of the council on medical devices, amending Directive 2001/83/EC, Regulation (EC) No. 178/2002 and Regulation (EC) No. 1223/2009 and repealing Council Directives 90/385/EEC and 93/42/EEC*", Chap. 5, Art. 51, pp. 49-50, 5 April 2017.
- [33] IEC. "*IEC 62304:2006 - Medical device software - Software life cycle processes*", Chap. 3, pp. 91-95, May 2006.
- [34] IEC. "*IEC 62304:2006 - Medical device software - Software life cycle processes*", "Introduction", pp. 87-89, May 2006.
- [35] ISO. "*ISO 14971:2007 - Medical devices - Application of risk management to medical devices*", Chap. 2, pp. 5-8, March 2007.

- [36] IEC. “*IEC 62304:2006 - Medical device software - Software life cycle processes*”, Art. 7, pp. 108-110, May 2006.
- [37] IEC. “*IEC 62304:2006 - Medical device software - Software life cycle processes*”, Annex A, pp. 114-116, May 2006.
- [38] Raissa Corallo. *Definizione di test e sviluppo della documentazione di validazione di un software medicale presso la fondazione CNAO*. Master’s thesis, Università degli studi di Pavia, Facoltà di Ingegneria, Dipartimento di ingegneria industriale e dell’informazione, A.Y. 2018/19.
- [39] IEC. “*IEC 62304:2006 - Medical device software - Software life cycle processes*”, Annex B, pp. 117-132, May 2006.
- [40] ISO. “*ISO 14971:2007 - Medical devices - Application of risk management to medical devices*”, Chap. 2, pp. 1-5, March 2007.
- [41] ISO. “*ISO 14971:2007 - Medical devices - Application of risk management to medical devices*”, Annex E, pp. 49-53, March 2007.
- [42] ISO. “*ISO 14971:2007 - Medical devices - Application of risk management to medical devices*”, Annex D, pp. 32-48, March 2007.
- [43] ISO. “*ISO 14971:2007 - Medical devices - Application of risk management to medical devices*”, Chap. 6, pp. 11-12, March 2007.
- [44] Quality-One. International Discover the Value. Failure mode and effects analysis (fmea). URL: <https://quality-one.com/fmea/>.
- [45] European Parliament and the council on medical devices. “*Regulation (EU) 2017/745 of the European parliament and of the council on medical devices, amending Directive 2001/83/EC, Regulation (EC) No. 178/2002 and Regulation (EC) No. 1223/2009 and repealing Council Directives 90/385/EEC and 93/42/EEC*”, Chap. 1, Art. 10, pp. 23-25, 5 April 2017.
- [46] Agile Coach Atlassian. *What is Agile?* URL: <https://www.atlassian.com/agile>.
- [47] Fondazione CNAO. Ambiente di sviluppo per la realizzazione di cyber-physical systems. *Unpublished document*. pages 1–6.
- [48] Paul Clements. *Software Product Lines - A New Paradigm for the New Century*. Software Engineering Institute, Feb. 1999, pp. 1-3.
- [49] Luigi Casalegno. *CF2020s: a model-driven service-oriented wizard-based multi-target development kit for supervision systems*. Project Book V2.1, Chapter 1, pp. 1-4.

- [50] JSON-Schema website. *JSON schema*. URL: <http://json-schema.org/>.
- [51] Atlassian Community. Requirements management: 6 best practices. URL: <https://community.atlassian.com/t5/Marketplace-Apps-Integrations/Requirements-management-6-best-practices/ba-p/742366>.
- [52] Vincenzo Alberto di Michele. *Waterfall Project Management vs. Agile Project Management: metodologie a confronto e implementazione nel campo aeronautico*. Master's thesis, Politecnico di Torino, Dipartimento di Ingegneria Meccanica e Aerospaziale. Corso di Laurea Magistrale in Ingegneria Aerospaziale, A.Y. 2020/21.
- [53] A. M. Madni and M. Sievers. "Model-based systems engineering: Motivation, current status, and research opportunities.", *21(3):172-190*, May 2018.
- [54] R. A. Sharples. "Implementation of Human System Integration (HSI) and 'Non-functional Characteristics' into the Systems Engineering Lifecycle - A Practical Approach at Airbus Defence and Space. *Procedia Manufacturing*", *21(3):172-190, 3:1896-1902*, 2015.
- [55] ISO. "ISO 14971:2007 - Medical devices - Application of risk management to medical devices", *Annex B, pp. 23-24*, March 2007.

A | Gantt Chart

The Fig.A.1 shows the entire list of operational activities carried out at the center in the six months.

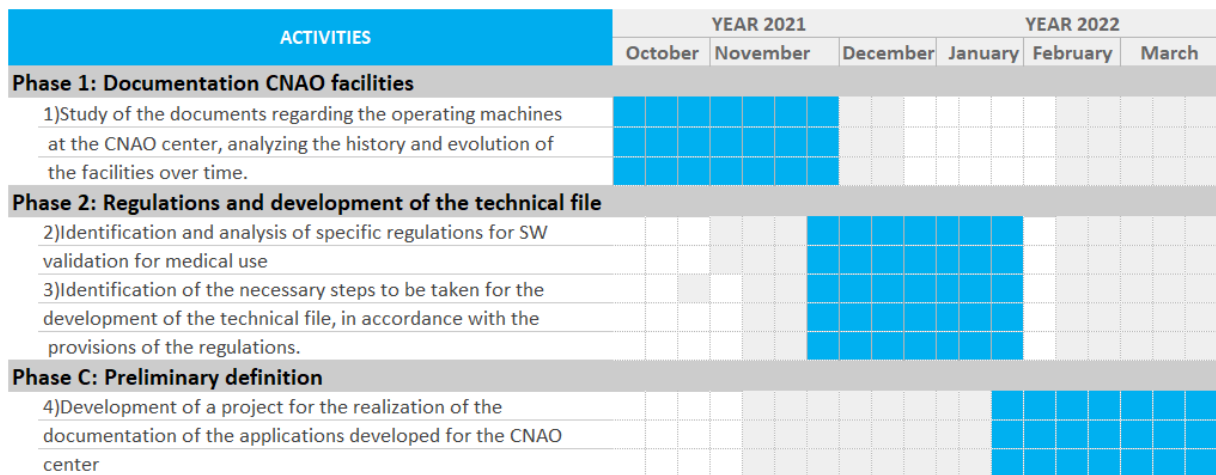


Figure A.1: Gantt Chart of CNAO center activities

B | System Architecture

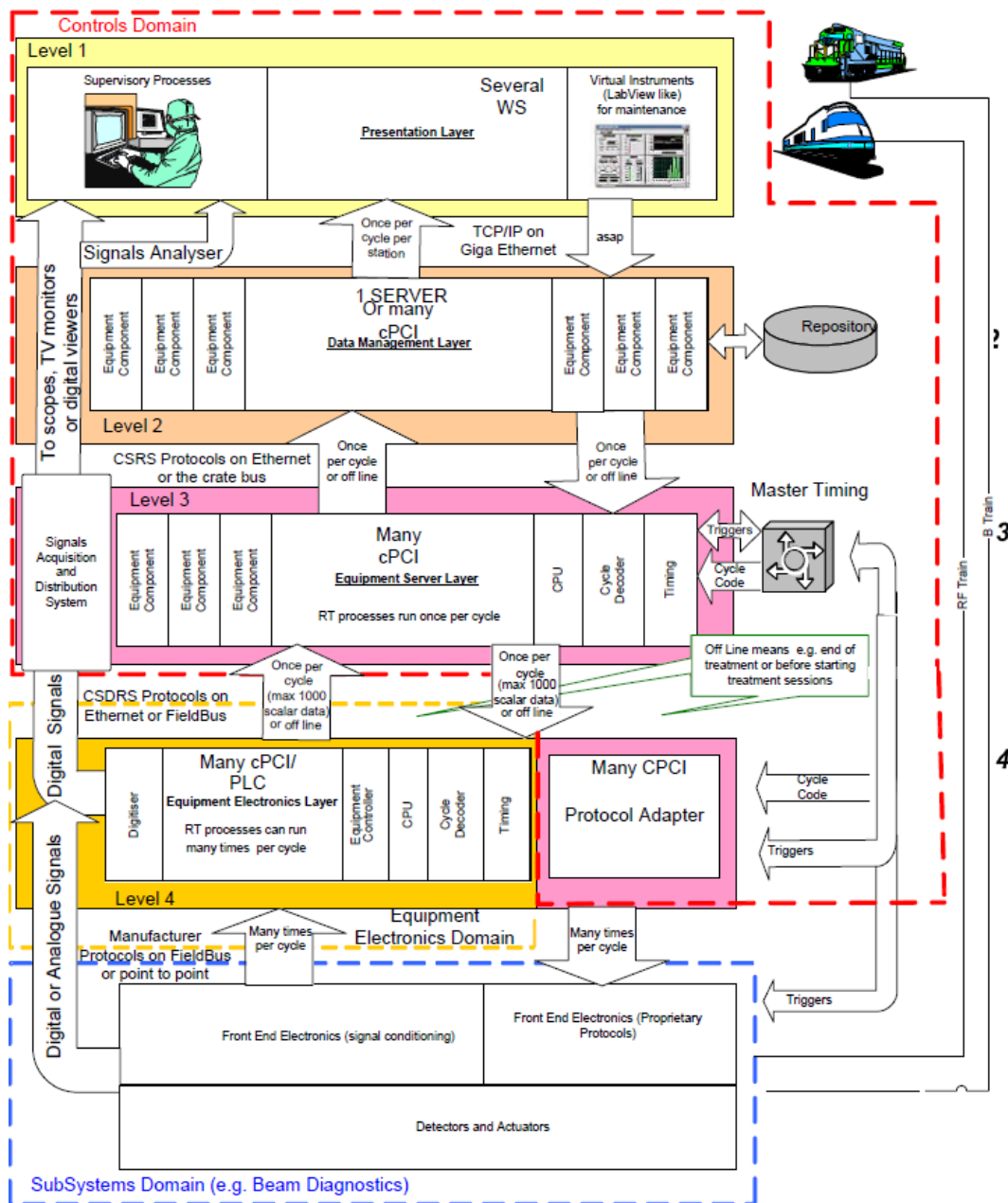


Figure B.1: Control System Layered Architecture[28]

C | Overview of the risk management process for medical devices

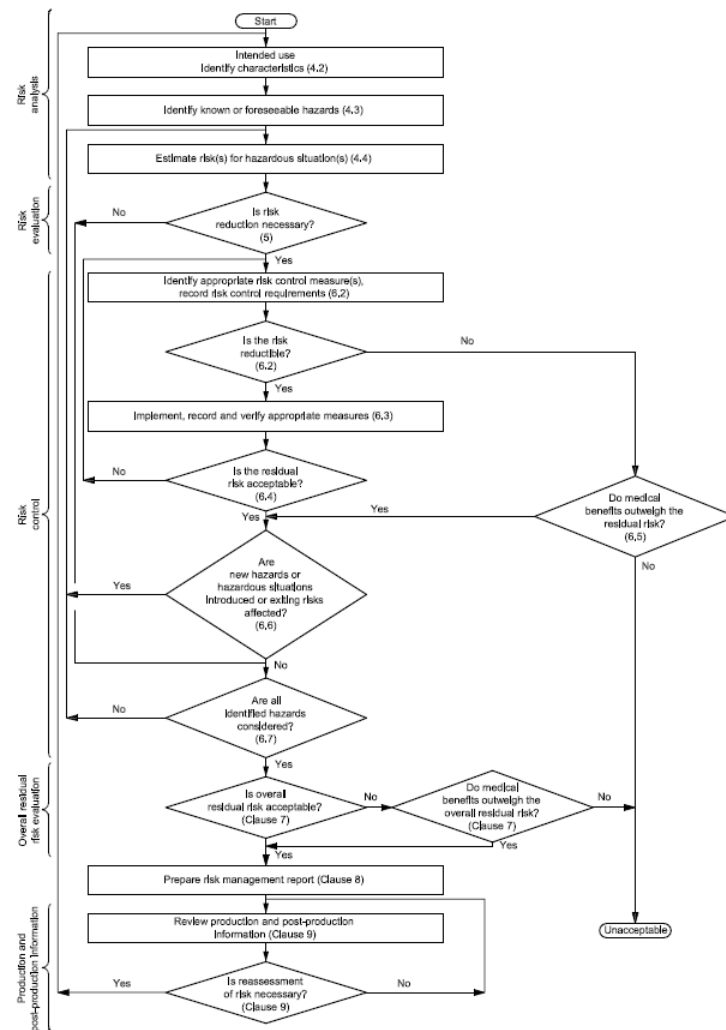


Figure C.1: Overview of risk management activities as applied to medical devices[55]

D | JSON Schema snippets

Definitions for CtrProjectDetail

```

"definitions": {
  "CtrTopicDetail": {
    "TopicName"
      The name of the Page or Topic
    "EpicName"
      Can contain the name of a group of pages that are logically related (Epic is a
      group of related Topics
    "TopicIcon"
      Can contain the name of an image file associated to the Page or Topic
    "EpicIcon"
      Can contain the name of an image file associated to the Epic
    "MainGridDataSource"
      Can contain the name of the database view that contains the list of the
      instances that go to fill the application Main Selection Page
    "MainGridFilterSource"
      Can contain the name of the database view that contains the filter for the
      instances that go to fill the application Main Selection Page
    "MainGridLayout"
      Can contain the names of the fields of MainGridDataSource that are not present
      in the application Main Selection Grid
      "$ref": "#/definitions/FieldType"
    "ISingleMode"
      For each page belonging to the Topic or for each Page in the Tabbed Page
      belonging to the Topic indicates if only one instance at a time can be edited
      (default true) or many instances can be edited in the same transaction",
    "ISingleInstance"
      For each Page belonging to the Topic or for each Page in the Tabbed Page
      belonging to the Topic indicates if only one instance at a time is collected
      (default true) when choosing from the Main Selection Grid or all the available
      instances are collected form the Data Source
    "Format"
      The format of the Main Selection Page
    "IsTabbedPage"
      True if the Page is a Tabbed Page",
    "PageType"
      Contains the type of the Pages belonging to the TemplateType. Depending on
      the TemplateType it can also be empty
    "UpperFormDataSource"
      Contains the name of the database table or view that are the Data Source used
      to fill the Pages belonging to the TemplateType
    "WhereCondition"
      Can contain a condition placed on the UpperFormDataSource to filter the data
    "Menus"
      Can contain the menu definitions that are assigned to the Pages belonging to
      the TemplateType
      "$ref": "#/definitions/MenuType"
    "ToolbarButtons"
      Can contain the toolbar buttons definitions that are assigned to the Pages
      belonging to the TemplateType
      "$ref": "#/definitions/ToolbarButtonType"
    "LowerFormDataSource"
      This is a Key/Value data pair in which the key is a string representing
      PageTitle and the Value are the database table or view names that are used to
      fill the detail pages of the TemplateType",
    "withUpperExtensionForm"
      For each Page having type EditGrid and DoubleGrid indicates if an Extension
      Page exists
    "withLowerExtensionForm"
      This is a Key/Value data pair in which the key is a string representing
      PageTitle and the Value indicates, for each Page having type DoubleGrid, if an
      Extension Page exists for the detail part of the Page
  }
}

```

```

"DetailPageType"
  This is a Key/Value data pair in which the Key is a string representing
  PageTitle and the Value, for each Page that is a Detail Page, indicates the type
  of the detail (grid, form or main form extension). This is only valid for pages
  having type MultiDetailPage
"IsDetailPage"
  When true indicates that the Page is a Detail Page in a Tabbed Page. The
  Position in the List corresponds to the position of the title of the page in
  PageTitle
"PageTitle"
  Contains the Title of the Page of the Topic or the Titles of the Pages when
  the PageType is Tabbed. The PageTitle must be unique in the application because
  it is used as a key for identifying the Page
"DetailTitle"
  This is a Key/Value data pair in which the Key is a string representing
  PageTitle of the master Page and Value can contain the Title of the Detail Pages
  in a MultiDetailPage
"IsDetailView"
  This is a Key/Value data pair in which the Key is a string representing
  PageTitle of the master Page and Value indicates if the data contained in each
  Detail Page depend on the content of the Master Page (true) or they are
  independent (false). This is only valid for pages having type MultiDetailPage
"IsDetailVisible"
  This is a Key/Value data pair in which the Key is a string representing
  PageTitle of the Master Page and Value indicates, for each Detail Page, if it is
  visible when the application starts (true). This is only valid for pages having
  type MultiDetailPage
"DetailChildNr"
  This is a Key/Value data pair in which the Key is a string representing
  PageTitle of the Master Page and Value are identifiers each Detail Page. This is
  only valid for pages having type MultiDetailPage
"WithExtensionPage"
  This is a Key/Value data pair in which the Key is a string representing
  PageTitle of the Master Page and Value indicates, for each Detail Page, if an
  Extension Page exists. This is only valid for pages having type MultiDetailPage
"FileNames"
  Reserved for future use
"RegisteredClasses"
  List of classes registered in the Unity Container
"OpcUaServers"
  For TemplateType CyberLine OPC-UA Servers Ids used in the application
"Reasons"
  Extra information that depends on the TemplateType
"PageLayouts"
  This is a Key/Value data pair in which the Key is a string representing
  PageTitle of the Page and Value indicates the layout of the Page ",
  "$ref": "#/definitions/PageLayout"
"required": ["TopicName", "Format", "IsTabbedPage", "PageType", "PageTitle",
  "RegisteredClasses"]
}
"ExtraPackage"{
  "PackageName"
    The name of the package or DLL
  "Version"
    The version of the package or DLL
  "IsDll"
    True if the package is a DLL
"required": ["PackageName", "Version", "IsDll"]
}
"ExtraXmlns" {
  "Name"
    The name of the xmlns reference
  "Namespace"

```

```

        The name of the referenced namespace
    "Assembly"
        The name of the assembly in which the namespace is contained
"required": ["Name", "Namespace", "Assembly"]
    }
"FieldType" {
    "FType"
        The type of the field represented by FieldType
    "FkTable"
        The object that contains the field represented by FieldType
    "FName"
        The name of the field represented by FieldType
    "IsRequired"
        True if the field represented by FieldType must have a value
    "IsPrimaryKey"
        True if the field represented by FieldType must have a value
    "IsIndicator"
        True if the field represented by FieldType must is read only
"required": ["FType", "FkTable", "FName"]
    }
"GraphInitializer" {
    "GraphId"
        Is an identifier of the graph in the application
    "Type"
        Defines the type of graph that can be X, Y graph(graph) or X, time graph
        (trend)",
    "MinY"
        Defines the minimum value of the Y coordinates
    "MaxY"
        Defines the maximum value of the Y coordinates
    "TimeSpan"
        Defines the amount of time to be shown in a trend graph
    "Title"
        Defines the title of the graph
    "GraphPosition"
        For future use: now it has a fixed default value
    "Panel"
        Contains the name of the pane in which the graph is placed. Allowed names are
        devicePanel followed by a number from 1 to 12 identifying the pane in the panel
        for master panel, setup panel, extension panel, extension detail
        panels;detailDevicePanel followed by a number from 1 to 12 for the first detail
        panel; detailDevice2Panel followed by a number from 1 to 12 for the first detail
        panel
    "PageType"
        Contains the name of the panel in which the graph is placed. In case of
        graphs placed in detail panels after the first detail the name of the panel is
        followed by a point(.) and a number identifying the detail panel. Panel names
        are composed of one of the words mastersnippet, detailsnippet, detail2snippet,
        setupsnippet, extensionsnippet, extensiondetailsnippet, extensiondetail2snippet
        followed by one of the words TopLeft, MidTopLeft, MidTopRight, TopRight,
        MiddleLeft, MidMiddleLeft, MidMiddleRight, MiddleRight, BottomLeft,
        MidBottomLeft, MidBottomRight, BottomRight
    "IsOpcUa"
        Contains true if the data source is an OPC UA server; false otherwise
    "HeightRequest"
        The height of the graph that defines also the height of the pane in case the
        pane is defined as being as high as its content (Auto)",
    "ExtraProperties"
        Defines all the optional extra properties affecting the graph layout with
        their values (see widget documentation for details",
"required": ["GraphId", "Type", "MinY", "MaxY", "Title", "GraphPosition", "Panel",
    "PageType", "IsOpcUa"]
    }

```



```

"GridInitializer" {
  "GridId"
    Is an identifier of the graph in the application
  "Title"
    Defines the title of the graph",
  "ClassText"
    Defines the code of the class used to retrieve the data that feed the grid",
  "GridPosition"
    For future use: now it has a fixed default value",
  "Type"
    The type of the source of data: Table when the data come from a database
    object; Device:<object class> when data come from an OPC-UA class",
  "PageType"
    Contains the name of the panel in which the graph is placed. In case of graphs
    placed in detail panels after the first detail the name of the panel is followed
    by a point(.) and a number identifying the detail panel. Panel names are
    composed of one of the words mastersnippet, detailsnippet, detail2snippet,
    setupsnippet, extensionsnippet, extensiondetailsnippet, extensiondetail2snippet
    followed by one of the words TopLeft, MidTopLeft, MidTopRight, TopRight,
    MiddleLeft, MidMiddleLeft, MidMiddleRight, MiddleRight, BottomLeft,
    MidBottomLeft, MidBottomRight, BottomRight
  "Panel"
    Contains the name of the pane in which the graph is placed. Allowed names are
    devicePanel followed by a number from 1 to 12 identifying the pane in the panel
    for master panel, setup panel, extension panel, extension detail panels;
    detailDevicePanel followed by a number from 1 to 12 for the first detail panel;
    detailDevice2Panel followed by a number from 1 to 12 for the first detail panel
  "Properties"
    Depending on the value of Type, the names of the database object columns or
    the names of the OPC-UA class properties from which the data are collected
  "PropertyTypes"
    The scalar type of each item of the Properties field
  "Devices"
    Depending on the value of Type, TABLE:<database object name> or the names of
    the OPC-UA class instances from which the data are collected
  "UncheckedFields"
    When the value of the Type field is Table, it contains the list of database
    object columns that are not shown in the grid
    "$ref": "#/definitions/FieldType"
  "HeightRequest"
    Contains the height of the grid that defines also the height of the pane in
    case the pane is defined as being as high as its content (Auto)",
  "ExtraProperties"
    Defines all the optional extra properties affecting the grid layout with their
    values (see widget documentation for details
    "$ref": "#/definitions/WidgetProperty"
  "ExtraColumnProperties"
    Defines all the optional extra properties affecting the grid columns layout
    with their values (see widget documentation for details",
    "$ref": "#/definitions/WidgetProperty"
  "ListLayout"
    Defines the same items present in the field Properties with extra information
    when using a Collection View instead of a data grid to present the collected
    data
    "$ref": "#/definitions/TagType"
  "required": ["GridId", "Title", "ClassText", "GridPosition", "Type", "PageType", "Panel",
    "Properties", "PropertyTypes", "Devices"]
}

"MenuType" {
  "Type"
    Indicates if the item is a menu header (menu) shown in the menu bar or a child
    of a menu (menuitem) shown when the menu is dropped down
  "Visible"

```



```

        True if the item is visible; false otherwise
    "Caption"
        The text that is shown in the suitable place when the item is visible
    "SubMenus"
        The list of menuitem instances that belong to this item and are shown when the
        item is dropped down
    "required": ["Type", "Visible", "Caption", "SubMenus"]
    }
    "PageLayout" {
        "Graphs"
            Can contain the definition of the Graphs present in the application user
            interface
            "$ref": "#/definitions/GraphInitializer"
        "Series"
            Can contain the definition of the data series instances contained in the
            Graphs present in the application user interface
            "$ref": "#/definitions/SeriesInitializer"
        "Grids"
            Can contain the definition of the Data Grids present in the application user
            interface
            "$ref": "#/definitions/GridInitializer"
        "masterPanels"
            This is a Key/Value data pair in which the Key is a string representing the
            pane name and the Value contains the definition of the visual elements present
            in the pane. Allowed key values are: mastersnippetTopLeft,
            mastersnippetMidTopLeft, mastersnippetMidTopRight ,mastersnippetTopRight,
            mastersnippetMiddleLeft, mastersnippetMidMiddleLeft,
            mastersnippetMidMiddleRight, mastersnippetMiddleRight, mastersnippetBottomLeft,
            mastersnippetMidBottomLeft, mastersnippetMidBottomRight,
            mastersnippetBottomRight
            "$ref": "#/definitions/TagType"
        "setupPanels"
            This is a Key/Value data pair in which the Key is a string representing the
            pane name and the Value contains the definition of the visual elements present
            in the pane. Allowed key values are: setupsnippetTopLeft,
            setupsnippetMidTopLeft, setupsnippetMidTopRight, setupsnippetTopRight,
            setupsnippetMiddleLeft, setupsnippetMidMiddleLeft, setupsnippetMidMiddleRight,
            setupsnippetMiddleRight, setupsnippetBottomLeft, setupsnippetMidBottomLeft,
            setupsnippetMidBottomRight, setupsnippetBottomRight
            ref": "#/definitions/TagType"
        "extensionPanels"
            This is a Key/Value data pair in which the Key is a string representing the
            pane name and the Value contains the definition of the visual elements present
            in the pane. Allowed key values are: extensionsnippetTopLeft,
            extensionsnippetMidTopLeft, extensionsnippetMidTopRight,
            extensionsnippetTopRight, extensionsnippetMiddleLeft,
            extensionsnippetMidMiddleLeft, extensionsnippetMidMiddleRight,
            extensionsnippetMiddleRight, extensionsnippetBottomLeft,
            extensionsnippetMidBottomLeft, extensionsnippetMidBottomRight,
            extensionsnippetBottomRight
            "$ref": "#/definitions/TagType"
        "detailPanels"
            This is a Key/Value data pair in which the Key is a string representing the
            pane name and the Value contains the definition of the visual elements present
            in the pane. Allowed key values are: (for the first detail page)
            detailsnippetTopLeft, detailsnippetMidTopLeft, detailsnippetMidTopRight,
            detailsnippetTopRight, detailsnippetMiddleLeft,
            detailsnippetMidMiddleLeft,detailsnippetMidMiddleRight,
            detailsnippetMiddleRight, detailsnippetBottomLeft, detailsnippetMidBottomLeft,
            detailsnippetMidBottomRight, detailsnippetBottomRight. (for the other detail
            pages) detail2snippetTopLeft, detail2snippetMidTopLeft,
            detail2snippetMidTopRight, detail2snippetTopRight, detail2snippetMiddleLeft,
            detail2snippetMidMiddleLeft, detail2snippetMidMiddleRight,

```

```

    detail2snippetMiddleRight, detail2snippetBottomLeft,
    detail2snippetMidBottomLeft, detail2snippetMidBottomRight,
    detail2snippetBottomRight
    "$ref": "#/definitions/TagType"
  "extensionDetailPanels"
    This is a Key/Value data pair in which the Key is a string representing the
    pane name and the Value contains the definition of the visual elements present
    in the pane. Allowed key values are: (for the first detail page)
    extensiondetailsnippetTopLeft, extensiondetailsnippetMidTopLeft,
    extensiondetailsnippetMidTopRight, extensiondetailsnippetTopRight,
    extensiondetailsnippetMiddleLeft, extensiondetailsnippetMidMiddleLeft,
    extensiondetailsnippetMidMiddleRight, extensiondetailsnippetMiddleRight,
    extensiondetailsnippetBottomLeft, extensiondetailsnippetMidBottomLeft,
    extensiondetailsnippetMidBottomRight, extensiondetailsnippetBottomRight. (for
    the other detail pages) extensiondetail2snippetTopLeft,
    extensiondetail2snippetMidTopLeft, extensiondetail2snippetMidTopRight,
    extensiondetail2snippetTopRight, extensiondetail2snippetMiddleLeft,
    extensiondetail2snippetMidMiddleLeft, extensiondetail2snippetMidMiddleRight,
    extensiondetail2snippetMiddleRight, extensiondetail2snippetBottomLeft,
    extensiondetail2snippetMidBottomLeft, extensiondetail2snippetMidBottomRight,
    extensiondetail2snippetBottomRight
    ref": "#/definitions/TagType"
  "MasterLayout"
    is the name of the layout chosen for the master panel: allowed values are
    Twelve,Eight,Nine, SixV, SixH, Four
  "SetupLayout"
    This is the name of the layout chosen for the setup panel: allowed values are
    Twelve,Eight,Nine, SixV, SixH, Four
  "ExtensionLayout"
    This is the name of the layout chosen for the extension panel: allowed values
    are Twelve,Eight,Nine, SixV, SixH, Four
  "DetailLayout"
    This is the name of the layout chosen for the detail panels: allowed values
    are Eight, SixH, Four",
  "ExtensionDetailLayout"
    This is the name of the layout chosen for the extension panel of the detail
    panels: allowed values are Twelve, Eight, Nine, SixV, SixH, Four",
  "MasterGraphs"
    This is a Key/Value data pair in which the Key is a string representing the
    pane name and the Value contains true if the pane contains a graph; false
    otherwise. Allowed key values are: mastersnippetTopLeft,
    mastersnippetMidTopLeft, mastersnippetMidTopRight, mastersnippetTopRight,
    mastersnippetMiddleLeft, mastersnippetMidMiddleLeft,mastersnippetMidMiddleRight,
    mastersnippetMiddleRight, mastersnippetBottomLeft, mastersnippetMidBottomLeft,
    mastersnippetMidBottomRight, mastersnippetBottomRight",
  "SetupGraphs"
    This is a Key/Value data pair in which the Key is a string representing the
    pane name and the Value contains true if the pane contains a graph; false
    otherwise. Allowed key values are: setupsnippetTopLeft, setupsnippetMidTopLeft,
    setupsnippetMidTopRight, setupsnippetTopRight, setupsnippetMiddleLeft,
    setupsnippetMidMiddleLeft,setupsnippetMidMiddleRight, setupsnippetMiddleRight,
    setupsnippetBottomLeft, setupsnippetMidBottomLeft, setupsnippetMidBottomRight,
    setupsnippetBottomRight",
  "ExtensionGraphs"
    This is a Key/Value data pair in which the Key is a string representing the
    pane name and the Value contains true if the pane contains a graph; false
    otherwise. Allowed key values are: extensionsnippetTopLeft,
    extensionsnippetMidTopLeft, extensionsnippetMidTopRight,
    extensionsnippetTopRight, extensionsnippetMiddleLeft,
    extensionsnippetMidMiddleLeft,extensionsnippetMidMiddleRight,
    extensionsnippetMiddleRight, extensionsnippetBottomLeft,
    extensionsnippetMidBottomLeft, extensionsnippetMidBottomRight,
    extensionsnippetBottomRight",
  "DetailGraphs"

```

This is a Key/Value data pair in which the Key is a string representing the pane name and the Value contains true if the pane contains a graph; false otherwise. Allowed key values are: (for the first detail page) detailsnippetTopLeft, detailsnippetMidTopLeft, detailsnippetMidTopRight, detailsnippetTopRight, detailsnippetMiddleLeft, detailsnippetMidMiddleLeft, detailsnippetMidMiddleRight, detailsnippetMiddleRight, detailsnippetBottomLeft, detailsnippetMidBottomLeft, detailsnippetMidBottomRight, detailsnippetBottomRight. (for the other detail pages) detail2snippetTopLeft, detail2snippetMidTopLeft, detail2snippetMidTopRight, detail2snippetTopRight, detail2snippetMiddleLeft, detail2snippetMidMiddleLeft, detail2snippetMidMiddleRight, detail2snippetMiddleRight, detail2snippetBottomLeft, detail2snippetMidBottomLeft, detail2snippetMidBottomRight, detail2snippetBottomRight",

"ExtensionDetailGraphs"

This is a Key/Value data pair in which the Key is a string representing the pane name and the Value contains true if the pane contains a graph; false otherwise. Allowed key values are: (for the first detail page) extensiondetailsnippetTopLeft, extensiondetailsnippetMidTopLeft, extensiondetailsnippetMidTopRight, extensiondetailsnippetTopRight, extensiondetailsnippetMiddleLeft, extensiondetailsnippetMidMiddleLeft, extensiondetailsnippetMidMiddleRight, extensiondetailsnippetMiddleRight, extensiondetailsnippetBottomLeft, extensiondetailsnippetMidBottomLeft, extensiondetailsnippetMidBottomRight, extensiondetailsnippetBottomRight. (for the other detail pages) extensiondetail2snippetTopLeft, extensiondetail2snippetMidTopLeft, extensiondetail2snippetMidTopRight, extensiondetail2snippetTopRight, extensiondetail2snippetMiddleLeft, extensiondetail2snippetMidMiddleLeft, extensiondetail2snippetMidMiddleRight, extensiondetail2snippetMiddleRight, extensiondetail2snippetBottomLeft, extensiondetail2snippetMidBottomLeft, extensiondetail2snippetMidBottomRight, extensiondetail2snippetBottomRight

"MasterGrids"

This is a Key/Value data pair in which the Key is a string representing the pane name and the Value contains true if the pane contains a data grid; false otherwise. Allowed key values are: mastersnippetTopLeft, mastersnippetMidTopLeft, mastersnippetMidTopRight, mastersnippetTopRight, mastersnippetMiddleLeft, mastersnippetMidMiddleLeft, mastersnippetMidMiddleRight, mastersnippetMiddleRight, mastersnippetBottomLeft, mastersnippetMidBottomLeft, mastersnippetMidBottomRight, mastersnippetBottomRight

"SetupGrids"

This is a Key/Value data pair in which the Key is a string representing the pane name and the Value contains true if the pane contains a data grid; false otherwise. Allowed key values are: setupsnippetTopLeft, setupsnippetMidTopLeft, setupsnippetMidTopRight, setupsnippetTopRight, setupsnippetMiddleLeft, setupsnippetMidMiddleLeft, setupsnippetMidMiddleRight, setupsnippetMiddleRight, setupsnippetBottomLeft, setupsnippetMidBottomLeft, setupsnippetMidBottomRight, setupsnippetBottomRight

"ExtensionGrids"

This is a Key/Value data pair in which the Key is a string representing the pane name and the Value contains if the pane contains a data grid; false otherwise. Allowed key values are: extensionsnippetTopLeft, extensionsnippetMidTopLeft, extensionsnippetMidTopRight, extensionsnippetTopRight, extensionsnippetMiddleLeft, extensionsnippetMidMiddleLeft, extensionsnippetMidMiddleRight, extensionsnippetMiddleRight, extensionsnippetBottomLeft, extensionsnippetMidBottomLeft, extensionsnippetMidBottomRight, extensionsnippetBottomRight",

"DetailGrids"

This is a Key/Value data pair in which the Key is a string representing the pane name and the Value contains true if the pane contains a data grid; false otherwise. Allowed key values are: (for the first detail page) detailsnippetTopLeft, detailsnippetMidTopLeft, detailsnippetMidTopRight, detailsnippetTopRight, detailsnippetMiddleLeft, detailsnippetMidMiddleLeft, detailsnippetMidMiddleRight, detailsnippetMiddleRight, detailsnippetBottomLeft, detailsnippetMidBottomLeft,

detailssnippetMidBottomRight, detailssnippetBottomRight. (for the other detail pages) detail2snippetTopLeft, detail2snippetMidTopLeft, detail2snippetMidTopRight, detail2snippetTopRight, detail2snippetMiddleLeft, detail2snippetMidMiddleLeft, detail2snippetMidMiddleRight, detail2snippetMiddleRight, detail2snippetBottomLeft, detail2snippetMidBottomLeft, detail2snippetMidBottomRight, detail2snippetBottomRight

"ExtensionDetailGrids"
 This is a Key/Value data pair in which the Key is a string representing the pane name and the Value contains true if the pane contains a data grid; false otherwise. Allowed key values are: (for the first detail page) extensiondetailsnippetTopLeft, extensiondetailsnippetMidTopLeft, extensiondetailsnippetMidTopRight, extensiondetailsnippetTopRight, extensiondetailsnippetMiddleLeft, extensiondetailsnippetMidMiddleLeft, extensiondetailsnippetMidMiddleRight, extensiondetailsnippetMiddleRight, extensiondetailsnippetBottomLeft, extensiondetailsnippetMidBottomLeft, extensiondetailsnippetMidBottomRight, extensiondetailsnippetBottomRight. (for the other detail pages) extensiondetail2snippetTopLeft, extensiondetail2snippetMidTopLeft, extensiondetail2snippetMidTopRight, extensiondetail2snippetTopRight, extensiondetail2snippetMiddleLeft, extensiondetail2snippetMidMiddleLeft, extensiondetail2snippetMidMiddleRight, extensiondetail2snippetBottomLeft, extensiondetail2snippetMidBottomLeft, extensiondetail2snippetMidBottomRight, extensiondetail2snippetBottomRight

"MasterZones"
 Contains a rectangle definition for each pane in the master panel: the rectangle defines the size of the pane

"SetupZones"
 Contains a rectangle definition for each pane in the setup panel: the rectangle defines the size of the pane

"ExtensionZones"
 Contains a rectangle definition for each pane in the extension panel: the rectangle defines the size of the pane

"DetailZones"
 Contains a rectangle definition for each pane in each detail panel: the rectangle defines the size of the pane

"ExtensionDetailZones"
 Contains a rectangle definition for each pane in each extension panel of each detail panel: the rectangle defines the size of the pane

"MasterGroupedZones"
 Defines the grouping of panes in the master panel: grouped panes are contained in a common frame instead of having a frame per pane

SetupGroupedZones"
 The grouping of panes in the setup panel: grouped panes are contained in a common frame instead of having a frame per pane

"ExtensionGroupedZones"
 Defines the grouping of panes in the extension panel: grouped panes are contained in a common frame instead of having a frame per pane

"DetailGroupedZones"
 Defines the grouping of panes in the detail panels: grouped panes are contained in a common frame instead of having a frame per pane

"ExtensionDetailGroupedZones"
 The grouping of panes in the extension panels of the detail panels: grouped panes are contained in a common frame instead of having a frame per pane

"MasterSizeLayout"
 Indicates if the row and columns of panes in the master panel have the size of the content (Auto) or use all the space available (Star). The string of values is comma separated

"SetupSizeLayout"
 if the row and columns of panes in the setup panel have the size of the content (Auto) or use all the space available (Star). The string of values is comma separated

"ExtensionSizeLayout"


```

        Indicates if the row and columns of panes in the extension panel have the size
        of the content (Auto) or use all the space available (Star). The string of
        values is comma separated
    "DetailSizeLayout"
        if the row and columns of panes in the detail panels have the size of the
        content (Auto) or use all the space available (Star). The strings of values are
        comma separated
    "ExtensionDetailSizeLayout": {
        Indicates if the row and columns of panes in the extension panels of the
        detail panels have the size of the content (Auto) or use all the space available
        (Star). The strings of values are comma separated
    "required": ["SetupGraphs", "SetupGrids"]
    }
    "SeriesInitializer" {
    "GraphId"
        The identifier of the graph to which the series instance belongs
    "Title"
        The Title for the series
    "BindableProperty"
        The MVVM property that is the data source of the series. In case of a property
        of an object the name is composed of the name of the object followed by an
        underscore (_) and the name of the property inside the object
    "Color"
        The Color used to draw the series data
    "Device"
        The name of the object from which the series data are taken
    "Index"
        The index of the series instance in the graph
    "ExtraProperties"
        Defines all the optional extra properties affecting the series layout with
        their values (see widget documentation for details
        "$ref": "#/definitions/widgetProperty"
    "required": ["GraphId", "Title", "BindableProperty", "Color", "Device", "Index"]
    }
    "TagType": {
    "Device"
        The name of the object supplying or presenting data
    "Property"
        The name of the property from which data is taken. The object presented by the
        TagType instance can also be unbound
    "ValueType"
        The type of the value represented; allowed values are: rostring, rodote,
        rodoube, roint, rouint, robool, robyte, rwstring, rwdote, rwdoube, rwint,
        rwuint, rwbool, rwbyte, rodoube[], roint[], rouint[], rwdoube[], rwint[],
        rwuint[], null, picture
    "LabelText"
        The text of the label associated to object being represented
    "LabelPosition"
        The position of the label associated to object being represented relative to
        the represented object; valid values are LEFT, TOP, RIGHT, LEFTFL, RIGHTFL,
        TOPFL, TOPC, NONE
    "WidgetType"
        The type of widget used to present the data
    "Category"
        When a combo-box is used to present the data, it contains the name of an
        enumeration of valid values to be assigned
    "WidgetLength"
        Length of the widget used
    "WidgetColumn"
        Widget in a pane can be grouped in columns. It contains the number of the
        column in which the widget defined by the TagType is placed
    "WidgetRow"

```

```

        It contains the number of the row in which the widget defined by the TagType
        is placed in a pane
    "WidgetHeight"
        The height of the widget in the pane
    "ImageSource"
        Can contain the name of an image that is contained in the widget defined by
        the Tag Type
    "ExtraProperties"
        Defines all the optional extra properties affecting the widget layout with
        their values (see widget documentation for details",
        "$ref": "#/definitions/WidgetProperty"
    "HasConverter"
        Can define the name of a converter (built in or custom) used to translate the
        source data before being presented and the presented data before being stored in
        the data sink
    "HasSnippet"{
        Can define an XAML snippet to be added to the widget definition in the XAML
        file
    "required": ["Device", "Property", "ValueType", "WidgetType", "WidgetLength",
        "WidgetColumn", "HasConverter", "HasSnippet"]
    }
    "ToolBarButtonType": {
        "Type"
            The type of the item. Contains a constant value; toolbarButton
        "Visible"
            True is the toolbar button is visible
        "Caption"
            The text contained in the toolbar button
        "Width"
            The width of the toolbar button
        "FontName"
            The name of the font used to draw the Caption
        "ColorName"
            The color used to fill the toolbar button background
    "required": ["Type", "Visible", "Caption", "Width", "FontName", "ColorName"]
    }
    "WidgetProperty"{
        "Name"
            The name of the property
        "Type"
            The type of the property (e.g., int, double, ...)
        "Text"
            The text to represent the property as an attribute in the XALM file
        "Description"
            A description of the meaning of the property
        "Value"
            A default value for the property the property ",
        "Group"
            Properties are placed in groups when presented in the widget property list. It
            contains the name of the group to which the property belongs ",
    "required": ["Name", "Type", "Group"]
    }
}
"properties" {
    "TemplateType"
        "The name of the model being used to build the application
        "enum": ["simple", "md", "tabbed", "epic", "ParallelCommands", "StateMachine",
        "CyberPanel", "CyberInstrument", "CyberGroup", "ListMonitor",
        "SchedulerMonitor", "DeviceMonitor", "Procedure", "CyberLine", "Spread"]
    "WLogin"
        The application needs a Login Page to accept user credentials",
    "WSettings"

```

```

    The application needs a Settings Page that should contain elements to
    configure the application (options for colors, optional confirmation for
    specific actions, ...)
  "WLoginInSettings"
    In the Settings Page fields to assign user credentials must be present
  "Schema"
    The name of the database from which application data are managed and OPC-UA,
    gRPC and OpenAPI connection addresses are retrieved
  "OpSystems"
    The operating system for which the application can be built are: Windows,
    Android, Ios, MacOS and Linux in this given order. A value of true in the List
    means that the application version for the corresponding Operating System must
    be built. At least one element must be true
  "FileName"
    The name of the directory in which the project for this application is kept
  "Topics"
    The description of the application contents",
    "$ref": "#/definitions/CtrTopicDetail"
  "Menus"
    Reserved for future use
    "$ref": "#/definitions/MenuType"
  "Reasons"
    The first element of the List can contain the image of the Main Selection Page
    when a Main Selection mechanism (data grid or file tree) is not present. The
    other elements are reserved for future use",
  "ExtraPackages"
    Contains the optional list of extra packages used by the developer by the
    developer to customize the application
    "$ref": "#/definitions/ExtraPackage"
  "Services"
    Can contain the List of OpenAPI or gRPC services that are used by the
    developer to customize the application
  "CustomXmlns"
    Can contain the List of extra namespace entries to be added to the XAML files,
    that define extra components used by the developer to customize the application
    "$ref": "#/definitions/ExtraXmlns"
  "AppResources"
    Can contain a XAML snippet that defines application-level resources used by
    the developer to customize the application
  "AppConfigurations"
    This is a Key/Value data pair that define application configuration values
  "required": ["TemplateType", "WLogin", "WSettings", "WLoginInSettings", "Schema",
    "OpSystems", "FileName", "Topics"]
}

```


List of Figures

1.1	Dose-depth curves of different beams[3]	6
1.2	Kuiper's <i>Standard Model</i> [15]	12
2.1	CNAO facilities[23]	19
2.2	LEBT layout[24]	20
2.3	MEBT layout[24]	22
2.4	General magnetic cycle trend[24]	23
2.5	Chopper layout[24]	27
2.6	Principle of active dose distribution[24]	28
2.7	Standard cycle chart displaying timing system events[27]	30
2.8	The main Control System Layers[28]	31
2.9	Control System Architecture[28]	32
2.10	Adapted conceptual architecture[29]	33
3.1	CE marking of conformity[31]	40
3.2	Overview of software development processes and activities[34]	42
3.3	Overview of software maintenance processes and activities[34]	42
3.4	A schematic representation of the risk management process[35]	47
3.5	Pictorial representation of the relationship of hazard, sequence of events, hazardous situation and harm[41]	49
3.6	Acceptability matrix	53
3.7	Index Risk Evaluation	53
4.1	Cyber-Physical System (CPS)[47]	60
4.2	Scheme to build a Product Line Control System	61
4.3	Model & Meta-Model	61
4.4	RTE scheme	62
4.5	Evolution in the Product Line	63
4.6	NGCS Architecture scheme	64
4.7	Toolkit structure	65
4.8	Documentation Construction Path	67

4.9	Process map without JSON Specification File	69
4.10	Complete process map	69
5.1	Example of the Dialog Display	74
5.2	Parallel Command Project Designer	75
5.3	Command Grid	76
5.4	Command Tree	76
5.5	Task Status	77
5.6	Task Utilities	77
5.7	End of Execution	78
5.8	Spread Project Designer	79
5.9	Data Spread Application	80
5.10	List Monitor Project Designer	81
5.11	Monitor Application	81
5.12	Scheduler Project Designer	82
5.13	Schedule Monitor	83
5.14	Editing Dialog Layout	84
5.15	Different types of graphs	85
5.16	Series instances chart values	86
5.17	State Machine Layout	88
5.18	Multi-Detail State Machine Layout	88
5.19	State Machine Data	89
5.20	Multi-Detail State Machine Data	89
5.21	Procedure Project Designer	90
5.22	First step example	91
5.23	Second step example	91
5.24	Jira Environment & Requirements	92
6.1	System of a vehicle used for re-entry from the atmosphere	96
6.2	V-Model scheme	97
6.3	Framework defined by Airbus DS[54]	101
A.1	Gantt Chart of CNAO center activities	111
B.1	Control System Layered Architecture[28]	113
C.1	Overview of risk management activities as applied to medical devices[55]	115

List of Tables

- 1.1 Carbon particle therapy facilities in clinical operation.[12] 10
- 3.1 Scores on severity and probability levels.[44] 52

Acknowledgements

First of all, I want to thank Prof. Giovanni Consolati, who showed great interest in my thesis from the beginning, providing me constant availability and attention. Thank you professor, because you have been of vital importance in finding this experience at the CNAO foundation, one of the most exciting experiences ever in my life.

In addition, I would like to thank the CNAO Foundation, with which I have been working for six months, the whole team of the Control Group, but above all my Company Supervisor, Dr. Luigi Casalegno, who gave me a great hand in the elaboration of the thesis and in learning new engineering software.

I thank my family, my Mom and Dad, who have always financed my studies and who have always given their best in guiding and advising me correctly. I hope Mom, I have made you happy with this work. Moreover, I hope that you will be always proud of me in your life. And I hope Dad, that this path of mine has shown you that I am able to face any challenge in any difficulty because I will never give up. Without all of you, I would have never made it!

Now, I would like to thank Eline, my big love, to whom I dedicate my thesis, who was able to turn my life upside down, helping me in moments of darkness and standing next to me in moments of joy, pampering me and making me smile. From the beginning you gave me advice on how to go forward, never resigning myself and warning me to remain calm and reduce anxiety, a component that has accompanied me in many moments of my life. I hope it's not my last official document to dedicate to you.

I wanted to thank my friends, to whom I owe them a lot, who accompanied me on this long MSc Degree.

And finally... I haven't forgotten about you, Roberto. Thanks also to you for making me the uncle of two extraordinary children twice, of Leonardo, my beautiful boy, and of Lavinia, my sweet little girl.

To all of you, Thank You!

