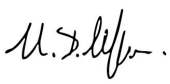
CTAO System Control Development Guidelines

Doc. No: CTA-TRE-SEI-000000-0017-1a

05 October 2022

	First/Last Name, Organisation, Role	Digital signature
Prepared by	E. Antolini, CTAO System Control Coordinator	Elisa Antolini Digitally signed by Elisa Antolini Date: 2022.10.18 15:58:22 +02'00'
Approved by	N. Whyborn, CTAO Lead SE	 Digitally signed by Nick Whyborn Date: 2022.10.19 09:42:47 +02'00'
Released by	W. Wild, CTAO Project Manager	Wolfgang Wild Digitally signed by Wolfgang Wild Date: 2022.10.31 14:03:42 +01'00'

Revision History				
Issue	**Rev.**	**Created**	**Reasons / Remarks / Section**	**Author**
1	a Draft01	2020.12.16	First draft creation	E. Antolini
1	a Draft02	2021.11.05	Content modified based on the Review process.	E. Antolini
1	a Draft03	2022.01.15	Content modified based on the Review process II iteration.	E. Antolini
1	a Draft04	2022.06.20	Content modified based on the IKC comments	E. Antolini
1	a Draft05	2022.07.07	Rewording of scope and minor edits	N. Whyborn / E. Antolini
1	a	2022.10.05	Final version agreed with IKC	E. Antolini

Authors	
First/Last Name, Organisation	**Contribution Subject/Chapter**
E. Antolini, CTAO PO	Producer of the document
I. Oya, CTAO PO	Managers Specification in Section 3.3, Section 4.2.3, 4.3.1, 5.2.5

Abbreviations	
ACADA	Array Control and Data Acquisition
ACS	Alma Common Software
AECS	Array Elements Control System
AIT	Assembly Integration and Test
BIT	Built In Test
CTA	Cherenkov Telescope Array
CTAO	Cherenkov Telescope Array Observatory
EUC	Equipment Under Control
FMEA	Failure Mode and Effect Analysis
HW	Hardware
ICT	Information and Communication Technology
IDL	Interface Definition Language
IM	Interface Manager
IP	Iterative Process
IPS	Integrated Protection System
IRS	Interface Requirements Specification
LCS	Local Control System
MOE	Measure of Effectiveness
MOP	Measure of Performances
PES	Programmable Electronic System
SRS	Software Requirements Specification
SW	Software

Table of Contents

Index of Figures

Index of Tables

1 Introduction

The control systems that should be provided for CTAO are defined in [AD-1]. Based on their functionalities, the different systems can be realized as industrial solutions, or off-the-shelf products. Among said systems, the Array Elements (AE) are the most critical and complex systems, since they are planned to be stand-alone controllable machines provided by different suppliers, that must be integrated into the centralized controllers (e.g., ACADA) [AD-1].

To transform efficiently the CTAO requirements into efficient control system solutions, easy to be integrated and maintained, the CTAO standards for the control systems [AD-2] require the application of the Life Cycle Processes for the development of the products. This document provides a possible tailoring on how to apply those technical processes, as intended in the international standards [RD-1] [RD-2], for the realization of the CTAO control systems.

The first part of this document shows how to apply the technical life cycle processes (Section 2) to ensure the development and maintenance of consistently high-quality products, throughout the construction stage and the whole lifetime of the observatory. The second part of the document describes with more details the application of the development processes to the CTAO customized products. The different layers of the system and the different technologies required are considered, to deliver a product that is compliant with the CTAO requirements, ready to be integrated in the observatory and to be maintained with reduced effort. Section 3 presents the specific set of requirements that should be derived for each control system; Section 4 is mostly dedicated to the application of the design specifications to the AECS but can be easily extended to every customized control system or integrated industrial solutions. Section 5 gives guidelines on how the Verification and Quality activities should be properly applied and integrated in the development process and scheduled to fit into the defined development phases.

1.1 Scope

The purpose of this document is to provide a possible interpretation of the engineering processes contained in the standards [RD-1][RD-2], to be applied for the realization of every control system provided for CTAO, to guarantee the delivery of a high-quality product in terms of efficiency, reliability, maintainability and cost saving [AD-2][AD-15].

Every section of this document should be considered as a guideline, and the application of the engineering activities proposed is not mandatory. It is not expected that every part of this document is applicable to every system. Each development team can take whatever is thought to be useful or helpful and customize the process based on the budget, needs and manpower, in order to address the quality requirements in [AD-2] and [AD-15].

This document is applicable to all CTAO contributors, private companies, or CTAO personnel responsible to deliver any control system defined in [AD-1], and intended to be designed and developed as products off the shelf (e.g. Array Element Control Systems) and/or integration of different industrial solutions (e.g. IPS). This is also applicable for future modification and/or upgrades of the operating systems.

The release of this document is not expected to be retroactively applied, especially where a system has passed CDR.

1.2 Applicable and reference documents

1.2.1 Applicable documents

AD-1 CTAO System Control Concept, CTA-TRE-SEI-000000-0016, Issue 1, Rev. a, 2022

AD-2 CTAO Control System Standards, CTA-STD-SEI-000000-0004, Issue 1, Rev. a, 2022

AD-3 Software Programming Standards, CTA-STD-OSO-000000-0001, Issue 1, Rev. a, 2020

AD-4 Interface Management Plan, CTA-STD-SEI-000000-0001, Issue 1, Rev. d, 2021

AD-5 CTAO Maintenance Concept CTA-TRE-SEI-312000-0001 (In prep.)

AD-6 CTAO Verification Management Plan (In Prep.)

AD-7 CTA Generic Telescope State Machine, CTA-SPE-ACD-000000-0001, Issue 2, Rev. f, 2020

AD-8 https://confluence.alma.cl/display/ICTACS/ACS+Documentation

AD-9 https://gitlab.cta-observatory.org/cta-computing/common/acada-array-elements

AD-10 https://confluence.alma.cl/display/ICTACS/ACS+Documentation

AD-11 https://gitlab.cta-observatory.org/cta-computing/common/acada-array-elements/dummy-telescope

AD-12 CTAO Glossary, CTA-LIS-INF-000000-001 (In prep)

AD-13 ICD for ACADA - Array Element Monitoring, CTA-ICD-SEI-000000-0004, Issue 1, Rev. b

AD-14 ACADA - Array Element Logging ICD, CTA-ICD-SEI-000000-0005, Issue 1, Rev. a

AD-15 Quality Plan, MAN-QA/110405, V 2.1, 2015

1.2.2 Reference documents

RD-1 ISO/IEC/IEEE 24748-4:2016 Systems and Software Engineering- Life Cycle Management -Part 4: Systems Engineering Planning.

RD-2 ISO/IEC/IEEE 15288:2015 Systems and Software Engineering- Life Cycle Processes.

RD-3 ACADA Quality Assurance Plan CTA-PLA-ACA-303000-0002, Is. 2, Rev. b

RD-4 CTA ACADA Software Development Life Cycle CTA-STD-ACA-303000-0001, Issue 2, Rev. d

RD-5 ACADA Validation, Verification and Quality Assurance Execution Plan CTA-PLA-ACA-303000-000, Issue 1, Rev. a

1.3 General Definitions

This document adopted the CTAO control system hierarchy and terminology defined in [AD-1]. Anyhow, for the purpose of this document and from the development point of view, the LCS are considered part of Programmable Electronic System (PES).

Programmable Electronic System: In the control environment, a PES is simply a generic term to indicate hardware and its direct controller for the purpose of control, protection, and monitoring (e.g. Motor + Drive + PLC, Interlocks + SafetyPLCs). We decided to use this term to be generic enough and to cover all the possible configurations adopted by the different CTAO control systems. As defined in [AD-1], the PES is a system based on a computer connected to sensors and/or actuators, interlocks, for the purpose of control, protection and/or monitoring. The PES can include various types of computers, programmable logic controllers, peripherals, interconnect systems, instrument distributed controllers, and other associated equipment.

Iterative Process: A process that provides the mechanisms for identifying and develop the product definitions of the system. This process is meant to be applied, throughout the system's life cycle to all activities associated with development, verification, and test.

System-Level: System is referring to the system of interest I.E. the product being developed.

2 Control System Engineering Life Cycle

Based on the standards provided by CTAO [AD-2] the development of the control systems at all levels of control for the hardware and software components, should be performed by applying the life cycle approach. The different systems should define their own number of phases based on the specific needs. Anyhow, the minimum number of stages that should be implemented and executed are shown in Figure 1.

System - level Definition	Sub-Systems Definition				System - Level Verification
	Preliminary Design	Detailed Design	Implementation	Assembly Integration and test (AIT)	

Figure 1: Control System Development Life Cycle.

As part of the development activity, a well-defined iterative and recursive process (Iterative Process) should be applied for each stage of the cycle, to produce a consistent set of requirements, functional arrangements, and design solutions.

Figure 2 shows a possible configuration of the iterative process. In this case it is divided in stages, applied sequentially top-down for each cycle, representing a valid mechanism that produce the defined deliverables (Outputs), considering the products of the previous stage of the cycle (Inputs) and providing input for the next level of development.

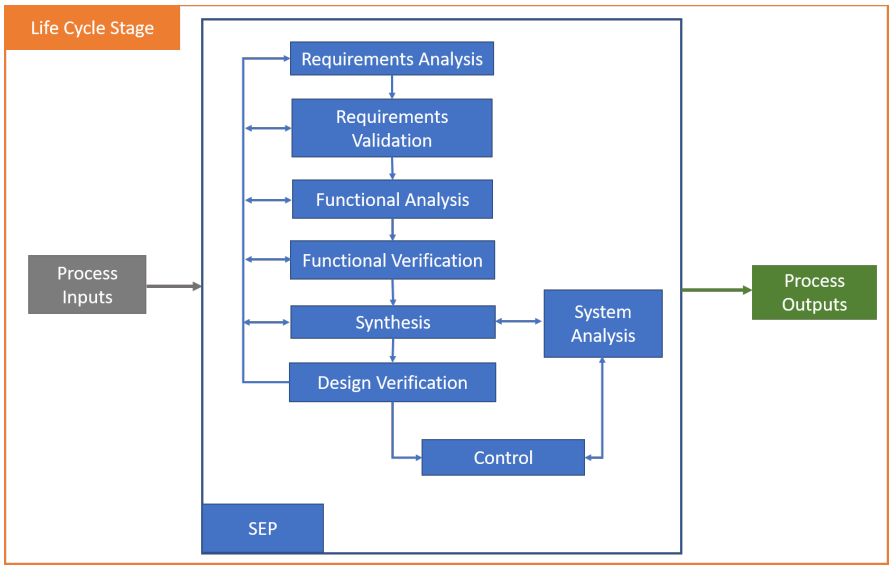


Figure 2: Iterative and Recursive process for the development life cycle.

The activities and outcome of the different stages of the development Life Cycle Process and the application of the iterative process to every phase of the cycle is presented below. The stages to be applied and their iterations vary based on the deliverables required for the different levels of development.

The *Control* phase is applied to every development level for the purpose of managing, documenting and change tracking of the development activity.

The tasks and activities of each stage of the iterative process are described in detail in Appendix.

2.1 System-Level Definition

This stage of the cycle establishes the definition of the control system with a focus on products at system level necessary to satisfy operational requirements. The documentation produced in this stage is required to guide developments at sub-systems level.

The specific activities to be accomplished are listed in Table 1.

Control System Definition		
Activity	**Description**	**Outcome**
Establish engineering and technical plans	• Criteria for determining system definition progress assessment • Allocation of technical resources among the engineering activities	• Engineering plan • Detailed schedule
Identify, Assess and Mitigate system risks (hardware and software)	• Determine all critical aspects of the system • Assess the identified security risks for critical assets • Define a mitigation approach and enforce security controls for each risk	• System risk Assessment and Mitigation analysis
Complete System specification (hardware and software)	• Define and control the system specification for each product of the system.	• System Specifications
Identify system and subsystems interfaces	• Subsystems identification • Define the design and functional interface requirements among the subsystems	• System Interface Specification • Sub-systems Interface Specification

	• Define the corresponding subsystem performance requirements and design constraints.	
Identify human/system interface	• Identify the interface requirements between humans (engineer/maintainer) and products or subsystems (performance, workloads, design constraints, and usability must be included).	• Human/system interface specifications • Maintenance/Training specifications

Table 1: System-Level Definition tasks

As shown in Figure 3, the iterative process is applied at this phase to generate system-level validated requirements baseline (Requirements Analysis and Validation stages), verified functional and design architectures, specifications, and system baseline (Functional Analysis and Verification stages) for the control system to be developed, starting from the CTAO high-level documentation as input.

The validated requirements baseline is documented in the integrated repository, and it is an input to functional analysis stage. The verified deliverables provided as output of the Functional Verification stage are used as input for the next stage of the cycle (Preliminary Design).

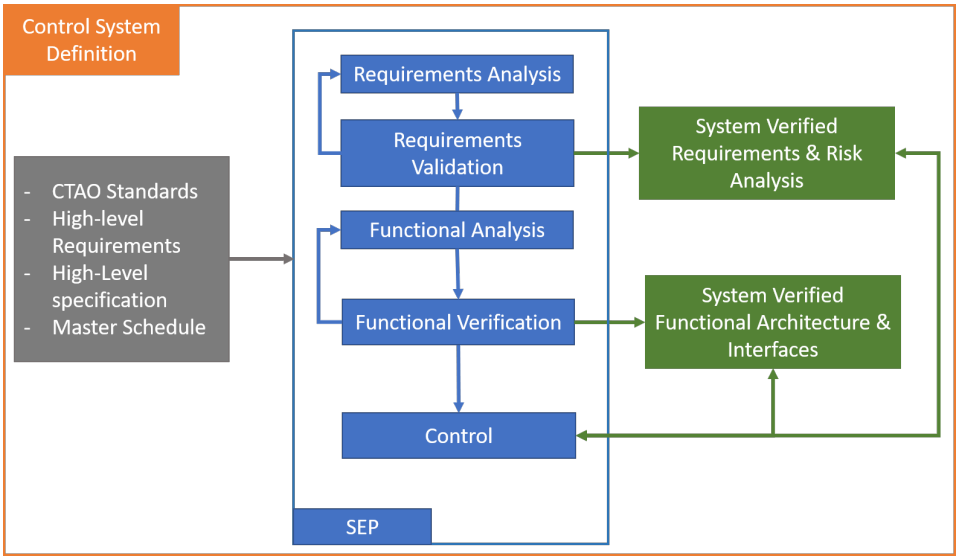


Figure 3: Iterative process for the System-Level Definition of the development life cycle.

2.1.1 Constraints

The product requirements are derived from stakeholder expectations, project constraints, external constraints, and higher-level system requirements.

2.1.2 Identify Variance and Conflicts

When voids in needs, constraints, etc., are identified or needs are not properly addressed, requirements analysis and validation are repeated until a valid requirements baseline is generated. When incompleteness is shown, functional analysis tasks are repeated to correct voids (see related *Identify variances and conflicts* tasks in Appendix).

2.1.3 Outcome - Deliverables

The following specifications at system level should be defined, properly documented, and placed under change control. Each specification should include a qualification section where the methods used to confirm that each system requirement has been satisfied under normal and abnormal conditions, are identified.

In constructing well-defined requirements care should be taken to establish agreed, non–ambiguous nomenclature and to identify in detail the fault conditions of the system.

System specifications

The system specifications should document the system requirements (functional and performance requirements, design characteristics, and design constraints) and the verification of the requirements from hardware and software point of view. The description of the software part of the system is covered by the Software Requirement Specification (SRS).

The definition of the system and qualification requirements must fulfill the applicable external requirement as defined in the relevant statement of work, requirement specification and IKC Agreement, as appropriate.

Software Requirements Specification (SRS)

SRS lays out functional and non-functional requirements, and must include a set of use cases that describe the interactions that the software must provide with the users and the other software. Software requirements specification permits a rigorous assessment of requirements before design can begin. It should also provide a realistic basis for estimating product costs, risks, and schedules.

System and subsystem interface specifications

The system interface specifications define the external functional and design interfaces for the system with respect to other systems (external interfaces). The subsystem interface specifications should define design and functional interface requirements among the subsystems (internal interfaces).

The definition of the external/internal interfaces must be compliant with the general guidelines for interface [AD-4] and the high-level architecture and must follow the standard defined [AD-2].

Human/system interface specifications

The team should define the interaction between humans and hardware and software elements identified in the control system design architecture.

Maintenance/Training specifications

The team should prepare a specification for the personnel required to maintain, and support the system throughout its life cycle, accordingly with the high-level specifications and standards provided by CTAO for maintenance [AD-5]. Analysis should determine the safety features required of the system and the level of training required for such personnel.

2.2 Preliminary Design

This stage initiates the subsystem design and the creation of sub-system level specifications and design to guide component development. Specific activities to be accomplished are listed in Table 2.

Preliminary Design		
Activity	**Description**	**Outcome**
Identify assemblies and assembly interfaces (hardware and software)	• Identify the assemblies of each subsystem • Define the design of each subsystem • Define functional interface requirements among the assemblies • Define the corresponding performance requirements and design constraints.	• Sub-systems and assembly specifications (hardware and software) • Up to date System specifications
Identify components and component interfaces	• Identify the components of each assembly • Define the design and functional interface requirements among the components • Define the corresponding performance requirements and design constraints.	• Component interface specifications (hardware and software)
Assess and mitigate subsystem risks	• Evaluate initial assets of the sub-systems and determine all critical assets. • Assess the identified security risks for critical assets • Define a mitigation approach and enforce security controls for each risk	• Sub-system risk assessment and mitigation analysis
Identify human/systems interface	• Identify the interface requirements between humans and assemblies or components.	• Up to date human/systems interface specifications • Up to date maintenance/training specifications

Table 2: Sub-System Definition- Preliminary Design

The Iterative Process at this stage is applied to each subsystem of the control system to be developed for the purpose of generating subsystem functional and physical architectures. Identified subsystem functions are decomposed into lower-level functions (assembly level) and allocating functional and performance requirements to component-level functional and physical architectures.

The verified functional architecture, appropriate to the level of development, is used in Synthesis and Design Verification to generate design solutions to satisfy stakeholder expectations as defined by the validated requirements baseline (see Figure 4).

The system analysis is required to evaluate the effectiveness of alternative design solutions, select the best design solution during synthesis and to manage risk factors throughout the systems engineering effort (see related tasks in Appendix).

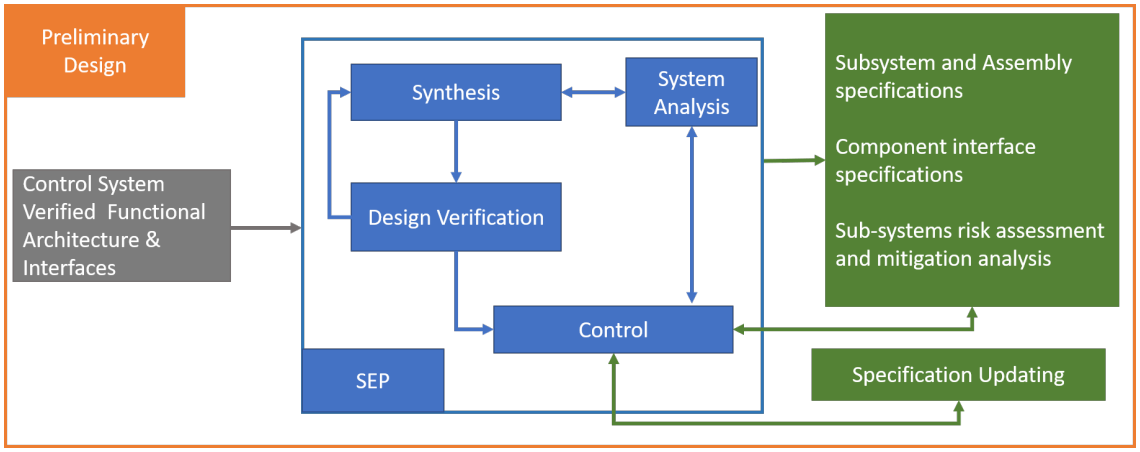


Figure 4: Iterative process application for the Preliminary Design stage of the life cycle.

The design architecture definitions and Specifications should be documented in the integrated repository, along with trade-off analysis results, design rationale, and key decisions to provide traceability of requirements up and down the architecture and used in the next application of the IP.

2.2.1 Constraints

The Sub-system and assembly specifications and interfaces must be compliant with the high-level requirements, architecture, specification, and standards provided by CTAO.

2.2.2 Identify Variance and Conflicts

When design architecture requirements are not traceable to the validated requirements baseline, it may require that synthesis be repeated to eliminate nonrequired functional and or performance requirements; or it may require that the IP activities be repeated to include those missing requirements (see related *Identify variances and conflicts* tasks in Appendix).

2.2.3 Outcome – Deliverables

The following specifications should be defined, properly documented, and placed under change control (Control stage of the IP).

The qualification section of individual subsystem specifications should identify the methods used to confirm that each subsystem requirement has been satisfied under normal and abnormal operating conditions.

The models, data files, and supporting documentation created during the design selection and verification should be maintained, and each version of a model or data file that impacts requirements, designs, or decisions should be saved in the integrated repository. Models may be digital, partial, or complete and may be hardware, software, or a combination of both, or may include human models or human-in-the-loop simulations or mock-ups for usability testing and workload measurement.

Sub-systems risk assessment and mitigation analysis

For critical sub-systems risks, simulations, scale model tests, or prototype tests should be used to demonstrate mitigation to an acceptable risk level with respect to cost, schedule, and/or performance.

Subsystem and Assembly specifications

These specifications should document the functional and performance requirements, design requirements or other imposed design constraints, and the qualification requirements for each subsystem.

The definition of the functional, performance and qualification requirements must fulfill the high-level requirement provided by the CTA project.

Component interface specifications

An interface specification for each component identified in the design architecture should be produced. Component interface specifications should document the functional and design interfaces among components, which should be satisfied by the selected design.

The definition of the component interfaces must be compliant with the general architecture provided by the CTA project and must follow the standard defined [AD-4].

Specification Updating

During this stage, all changes to established specifications under control should be updated:

- System and sub-subsystems interface specification
- System specification
- Validation, Verification and Quality assurance plans
- Human/system interface specifications
- Maintenance/Training specifications

2.3 Detailed Design

This stage completes the subsystem design down to the lowest component level and create a specification for each component. The outputs of this stage are used to guide fabrication of preproduction prototypes for development testing. Specific activities to be accomplished are listed in Table 3.

Detailed Design		
Activity	**Description**	**Outcome**
Complete component definition (hardware and software)	• Decompose the components of each assembly to a level sufficient for design completeness • Complete the definition of each subcomponent and component • Control the interfaces among the subcomponents.	• Component specifications (for hardware and software) • Up to date system, subsystem, and assembly specifications • Up to date human/system interface specifications • Up to date maintenance/training specifications
Prepare integrated data package	• Detailed documentation for each component and its subcomponents to satisfy functional architecture requirements, component interface specifications, and the assembly specification.	
Address component risks	• Mitigate component-level risks that were assessed to be critical to component development during subsystem definition.	• Component risk assessment and mitigation analysis

Table 3: Sub-System Definition- Detailed Design.

The Iterative Process is applied to each component and its subcomponents for the purpose of generating component functional and design architectures (see Figure 5). Identified component functions are decomposed into lower-level functions, and functional and performance requirements are allocated throughout the resulting lower-level functional and design architectures.

In this stage the subsystem requirements are fully defined.

The verified design architecture is used to form the specification tree for the system, and when combined with the verified life cycle process design architectures, forms the system architecture.

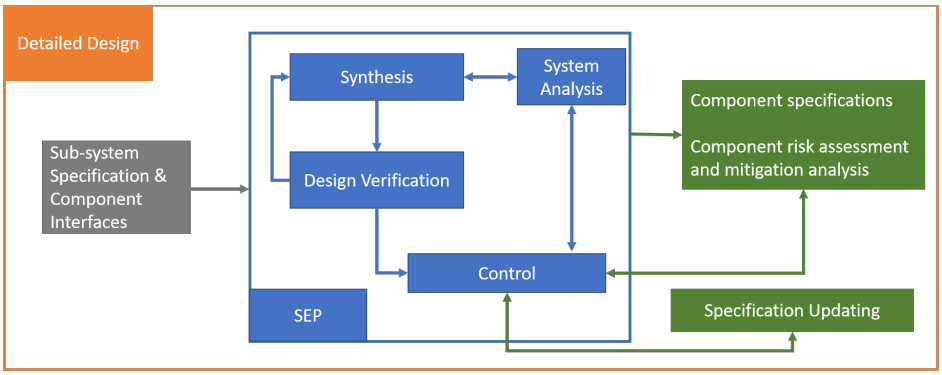


Figure 5: Iterative process application for the Detailed Design stage of the life cycle.

2.3.1 Constraints

Component specifications must be compliant and traceable with the applicable requirements, architecture, specification, and standards.

2.3.2 Identify Variance and Conflicts

See 5.2.2 and apply at component-level.

2.3.3 Outcome – Deliverables

The following specifications should be defined and properly documented. The qualification section of individual specifications should help identify the methods that will be used to confirm that each component requirement has been satisfied under normal and abnormal operational conditions.

The models, data files, and supporting documentation created during the design selection and verification should be maintained, and each version of a model or data file that impacts requirements, designs, or decisions should be saved in the integrated repository. Models may be digital, partial, or complete and may be hardware, software, or a combination of both, or may include human models or human-in-the-loop simulations or mock-ups for usability testing and workload measurement.

Component specifications

The project should complete a specification for each component included in the design architecture, which identifies the functional, performance and qualification requirements for the component. Internally approved, Verification and Quality assurance plans should be provided.

Specification Updating

During this stage, all changes to established specifications under control should be applied:

- System, product, subsystem, and assembly specifications
- Human/system interface specifications
- Maintenance/Training specifications

Component risk assessment and mitigation analysis

For critical component risks, simulations, scale model tests, or prototype tests should be used to demonstrate mitigation to an acceptable risk level with respect to cost, schedule, and/or performance.

2.4 Implementation

The purpose of this stage is to perform the implementation of the software and the procurement of the hardware based on the approved specification produced from the previous stages.

2.5 Assembly, Integration and Test

The purpose of the AIT stage is to assemble and integrate the system (hardware, software, and users), meanwhile continuing to develop confidence that it will be able to meet the system requirement and satisfy the specifications.

The major activities of this stage are:

- Assemble, integrate, and test components and assemblies (hardware and software)
- Assemble, integrate, and test subsystems (hardware and software)
- Update and control all changes to approved specifications (if necessary).

The integration activities ensure that combining the lower-level elements results in a functioning and unified higher-level element, with logical and design interfaces satisfied. Subcomponents must be progressively assembled and integrated into complete components, components into assemblies, assemblies into subsystems, subsystems into a complete system.

The test activities should verify that the system will meet system requirements, by first testing the components and then conducting tests at each level up to the total system. At each level of assembly and integration, the components, assemblies, subsystems, and system should be subjected to sufficient testing to ensure operational effectiveness, usability, interface conformance, conformance with specified requirements, producibility, and supportability.

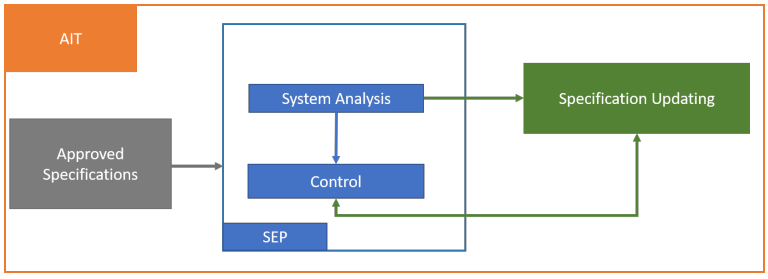


Figure 6: Iterative Process application for the Detailed Design stage of the life cycle.

The Iterative Process is applied during this stage for the purpose of resolving product deficiencies when specifications for the system, subsystem, assembly, or component are not met, as determined by inspection, analysis, demonstration, or test (See Figure 6).

At the end of the AIT activity the following deliverables should be produced and place under control (Control stage of the Iterative Process):

- Up to date and approved specifications (If necessary)

- Components, Assembly, sub-systems test reports
- Integration Test Reports
- Qualification Test Reports

2.6 System Verification Stage

The verification procedure evaluates whether the control system complies with the requirements and specifications imposed during the development phase.

The procedure involves the organization of internal audits (see Section 5.1) to demonstrate that the system has been verified to satisfy specification and baseline requirements for each system level and to confirm readiness for validation stage.

- Issues for the components, assemblies, subsystems, products, and life cycle processes and services are resolved.
- Test procedures for components, assemblies, subsystems, and system were completed and were accurate.
- Tests were conducted in accordance with established procedures.
- All component, subsystem, and system products meet specification requirements.
- Risk-handling procedures are satisfactory.

The result of this activity should be properly documented with a dedicated report (e.g. Verification Report)

2.6.1 Non-Compliance

Any non-compliance should follow the formal procedure defined by CTAO [AD-15].

Anyhow, as general advice, when a subcomponent, component, assembly, subsystem, or system fails to satisfy its requirements, the deficiency should be analyzed to determine the cause of the problem and apply the System Engineering Process to resolve the problem. The product should then be retested to ensure the compliance with specified requirements.

2.7 Tools and Methodologies

All the activities related to the engineering process should be performed using adequate analysis tools and methodologies for:

- Numerical studies to define a control architecture compliant with requirements (considering constraints such as cost, schedule, and risk)
- Numerical Analysis supporting the control detail design
- Performance verification analysis (including simulation)
- Evaluation of Code quality and verification [RD-3][RD-4][RD-5]

The assessment of safety hazard must follow the standards provided in [AD-2]. Some technique that can be used is listed below:

- Process Hazard Analysis (PHA)
- Fault Tree Analysis (FTA)
- Hazard and Operability Study (HAZOP)
- Failure Mode and Effect Analysis (FMEA)
- Event Tree Analysis (ETA)
- Quantitative Risk Assessment (QRA)

2.7.1 Software Tools

The analysis should be performed with well recognized software packages. For instance, MATLAB/Simulink are the recommended software packages to perform servo control loop simulations.

For the required calculations, such as failure rate and reliability, information must be collected from suppliers for specific components or using failure rate from similar applications from the past experience or specialized prediction software (e.g. Reliasoft).

A suitable set of tools, including languages, (graphical) editors, compilers, and configuration tools, should be selected for the required integrity level of the software [AD-2].

To determine the Performance Level (PL) of safety related control systems, the free tool Performance Level Calculation is recommended by ISO 13849-1.

2.7.2 SW tools for AE Managers

Based on the software languages and development environment defined for AE implementation (see 4.2.2), two types of tools are recommended to perform the automated code analysis and the Unit Testing:

- SonarQube, which is an open-source platform for ensuring code quality. [https://www.sonarqube.org/]

- Jenkins, which is a self-contained, open-source automation server which can be used to automate all sort of tasks related to building, testing and delivering software [https://www.jenkins.io/]

3 General Requirements for Control Systems

This Chapter provides guidelines on how to define the internal requirements specification in the context of developing the control system of the corresponding system, subsystem or instrument, as grouped in [AD1].

Whenever the final product is realized through the presence of devices already available as industrial solutions (e.g. IPS), the requirements to be defined must be focused to led to the right choice of the devices, in a way that the technical features fulfill the project needs, and to ensure the integration and verification of the final product.

3.1 PES Requirement Specification

This section described the requirements needed for the safety and non-safety CTAO control systems to be developed as PES. For the AECS the PES includes the LCS [AD-1].

For the scope of this document the operators of the PES are the Expert Operators, as defined in [AD-12], and the actors can be all the external Software which interacts with the software of the PES.

In the requirement specification phase, it should be stated clearly and in the appropriate detail, what the Control System has to do with the system or process, distinguishing within the document the non-Safety related requirement specification, applied to the Non-Safety Related PES, from the Safety related requirement specification, applied to the Safety-related PES, in the sense specified in the related standards [AD-2].

The Safety and Non-Safety Requirements Specification should comprise:

- The *Functional Safety (non-Safety) Requirements Specifications*, that should contain all the requirements necessary for the design phase to achieve the required functional operational of the PES.

- The *Safety Integrity Requirements Specifications,* that should include modes of expression and descriptions which are understandable and appropriate to the duties they must perform (to personnel involved in the design) operation and maintenance of the PES.

The Safety and Non-Safety Requirements Specification should be expressed and structured in such a way that it is clear, precise, unequivocal, verifiable, testable, maintainable, and feasible.

The most important categories that need to be considered while defining PES requirements are provided below.

Environment

Consider the environment in which the PES operates.

Modes of operation

Describe the modes of operations of the process including the operator responsibilities in the process.

Control/Safety system behaviour

Describe and document the required behaviour of the safety (and non-safety) systems (e.g. Use Cases at LCS level), in normal process operation and in the presence of potential safety hazards processes.

Global Hazard analysis

Include the control system in the Hazard analysis of the system, identifying the constraints on the control system part and, on rest of equipment and process.

Risk classification and risk reduction

Identify the risk class (applying the risk graph from relevant standards) and define the system integrity levels.

Process behaviour and interface

Specify all aspects of the process that can be protected or monitored by the PES:

- Describe the detailed behaviour of the process during normal operation, in the presence of failures and in any other operating mode such as maintenance or calibration activities.
- Specify in detail the interfaces (include also physical and electrical levels) between the PES and the process.

Operator role and interface

Define all mode of communication between the operator/actors and the PES:

- Define the behaviour of the operator, for correct operation of the System, considering the possible impact of operator failure, and the consequent impact on System safety
- Consider the *Identification* & *Authentication,* as well as *Access Control* of the operator. The security log should record the operator actions.

Functionality of the PES

Specify the expected functional behaviour of the PES (safety and non-safety related) to take account of both normal operation and operation with failures. It should be considered that, under any event or any combination of events so described, the system as a whole, will remain in or move to a safe state.

Integration Plan

Procedures for system integration, system test, assessment and final acceptance should be clearly documented including a detailed description of responsibilities between customer, supplier, and assessor.

Verification Matrix

Define a Verification matrix, listing the requirements contained in the specification document and identifying for which milestone (e.g. gateway reviews), and according to which method, these shall be verified.

Procurement Planning

All requirement, specification, application software development, integration, installation testing, certification, and acceptance carried out internally by the development teams must be properly documented [AD-2].

Performance

It is important that overall timing process characteristics are clearly documented, and that the timing and synchronization of the PES meets the timing needs. Integrity levels, as intended in the related standards [AD-2], for both hardware and software components should be identified both in normal system operation and in the presence of external failures, to which, the PES should respond. Mechanisms for diagnosing problems and monitoring the operation of the PES should be described.

Maintenance requirements

Mechanisms for maintenance and, replacement of components and systems (including software), should be documented. The effect of invoking these on the whole System should also be described.

3.1.1 Non-Safety requirement specifications

The objective is to develop the PES Requirement Specification for the control systems necessary to implement the non-safety related operational functions required by the system. The non-safety related requirements specification applies to the Control Units of the PES, as defined in [AD-1].

The *Functional non-Safety Requirements Specification* depends heavily on the type of product that is being developed and its purpose. Anyhow, the following list of key-field should guide the contributor to come to a complete specification.

Key-Field	Dependencies and Contents
Size of data to be controlled	Numbers and types of sensors (e.g., precision, resolution) Numbers and type of actuators (e.g., precision, resolution) Timing requirements
Technical HMIs	Interfaces for operators Monitors, keyboards, mouse touchscreen Access rights for operators Uniform I/O to the user Programming environment [AD-2] Use of standard software modules [AD-2] Change management (on-line or off-line) Graphical/textual input Communication between processes and engineering system Use of standards [AD-2] National language guidelines and documentation [AD-2]
Report of the system	Time stamps Ordering or sequence of events System wide/single clock and related accuracy

Table 4: Control System Definition tasks

After the analysis, each feature must at least fall into one of the following categories: Mandatory, Optional, Not Required, Not applicable.

3.1.2 Safety requirement specification

The objective is to develop the function requirements Specification for the safety related control systems necessary to implement the required safety functions (see Section 4.1.5).

The requirements specification of this category applies to the Safety Units of the PES, as defined in [AD-1], and must follow the standards provided for Safety [AD-2].

Below is reported a summary of the main aspects that the functional Safety Requirements Specification must cover:

- Identify the required safety functions in order to achieve functional safety.
- Define the safety-related equipment to implement the safety functions.
- Analyse throughput and performance response time.
- Define system and operator interfaces.
- Define interfaces between the safety-related system and any other systems.
- Consider any safety relevant information which may have an influence on the safety-related system design.
- Consider all relevant modes of operation of the EUC.

- Consider all required modes of behaviour of the safety-related system. In particular, failure behaviour and the required response of the safety-related system should be detailed.
- Identify and document any relevant constraints between the hardware and the software should be identified and documented.
- Include all environmental conditions which are necessary to achieve functional safety.
- Define the procedures for starting up and restarting the PES.
- Specify requirements necessary to enable monitoring of the PES hardware to be undertaken.
- Individuate requirements for periodic testing of the safety functions.

Where the PES safety-related system must implement both safety and non-safety functions then the hardware and software should be treated as safety-related unless, it can be shown that there is adequate independence between the safety and non-safety functions.

Relaxation from this requirement should only be permissible if it can be shown that:

- The safety functions are independent.
- The implementation is independent.
- The failure of any non-safety-related functions does not affect the safety-related functions.

3.1.3 Safety integrity requirement specification

The objective is to develop the Safety Integrity Requirements Specification for each of the safety functions, considering the assigned Integrity Level as intended in the related standard [AD-2].

Where the PES must implement safety functions of different Safety Integrity Levels then the PLC logic solver (hardware and software) should be treated as belonging to the highest Safety Integrity Level. The requirements for safety integrity, as they relate to "the control of errors" in the PES design, should be composed of requirements for:

- hardware integrity.
- software integrity.
- system integrity comprising environment and operation.
- data integrity and application (software) integrity checks.

The requirements for the above error clauses are discussed in the related standards reported in [AD-2].

3.1.4 State Machine

The *Deterministic Finite State Machine* approach, as defined in [AD1], is mandatory for every layer of CTAO control software, as stated in [AD-1]. Based on that, the Safety and non-Safety PES requirement specification must include the definition of a dedicated state machine for every layer of software included in the Safety and non-Safety PES.

The deterministic state machine presented here must be intended as a possible, generic, and not unique representation of the state machines that ca be defined for every LCU of the CTAO Array Elements.

The number of states, the names, the state transitions, and the specific configurations can be customized as needed, as much as the deterministic aspect of the system is respected: undefined states and unknown configuration cannot be accepted [AD-1].

In this section a simple classification of a system into (at least) 5 states and their possible transitions is proposed. A sketch of the state machine is reported in Figure 7.

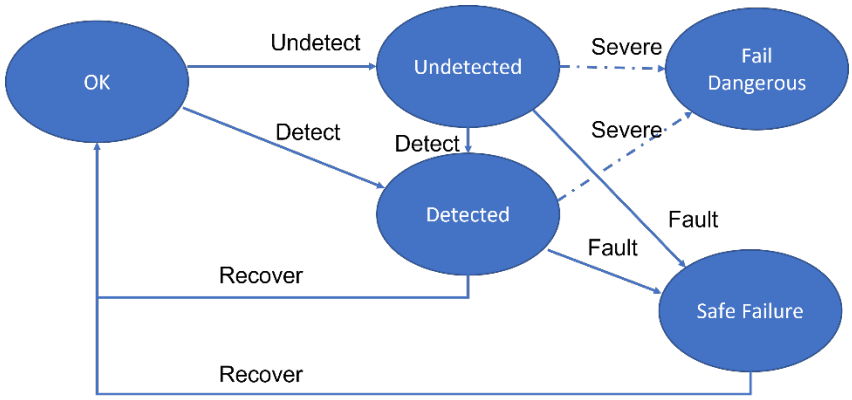


Figure 7: PLC States and Transitions

The states defined are:

- *OK,* where the system is operating normally.
- *Undetected*, in which the system is operating with an undetected Error/Failure condition.
- *Detected*, in which the system is operating, the error has been detected and the acknowledge of the error has been done. In this status the fault condition has been classified and recognized and the system is operating. The aim is to provide an adequate level of assurance that a particular fault will be discovered depending on the severity of the fault.
- *Safe Failure* is the status in which the system has failed safely. The system is not operating but can be recovered. Those faults which require (and possess) recovery actions must be identified.
- *Fail Dangerous* is the status in which the system has failed in a dangerous manner. The system is not operating. The development process should guarantee the absence of the transitions to this state.

Since the document applies to different systems, with different functionalities, we decided not to give any further advice for the states to be defined during operations, which in this scheme is generically indicated as “OK”. The fault management instead can have the same states and transitions, despite the functionalities of the systems being different.

The permitted transitions are:

- *Recover*, which takes the PLC system to the *OK* status after the recovery from a failure situation (*Detected*, *Fail Safe*).
- *Detect*, which takes the system to *Detected* status after the occurrence of an error/failure situation that has been identified.
- *Undetect*, which takes the system to *Undetected* status after the occurrence of an error/failure situation that has not been identified. Such a transition can arise for a number of different reasons which typically need some analyses that should be carried out during

systems requirements analysis. The state transition into the Fail Safe is mandatory if the cause of the error is not promptly identified.
- *Fault*, which takes PLC system to *Fail Safe* status. These failures do not influence the safety of the system, but interrupt the operation of the equipment under control in a safe way.
- *Severe*, this transition takes the system directly to the *Fail Dangerous* status. Such failures constitute a major threat to the integrity of the system. In devising the PLC requirements, the development process should guarantee the absence of such transitions.

From this classification the appropriate state machine related to the specific system can be built. The functional and non-functional requirements for the state transitions should be implemented following the engineering practices for the development of the PES systems presented in this document.

3.2 PES Interface Requirements

The Interface Requirements Specification (IRS) specifies the requirements imposed on the hardware and software components of the PES to achieve one or more interfaces among these entities.

The IRS can be used to supplement the System Requirements Specification as the basis for design and qualification testing of systems and the related hardware and software components.

In defining the PES interface requirements, the standard communication protocols provided by CTAO between the following systems must be considered [AD-2]:

- Communication protocols between field devices and the supervisors (e.g. Profinet, EtherCAT, Profibus).
- Communication protocols between Supervisors and Manager (e.g. Ethernet)
- Communication protocols between IPS and Safety Control Units of CTAO (TBD)
- Communication protocols between IPS and ACADA Monitoring (See Section 3.4)
- Communication protocols between Power Management System and ACADA Monitoring (See Section 3.4)
- Communication protocols between Power Management System and IPS (TBD)
- Communication protocols between ICT and ACADA Monitoring (See Section 3.4)

Notes:

- The technical feature to define the protocols for the monitoring of ICT, PMS and storage of IPS parameters are still under investigation.
- The needs for direct connection between IPS and the safety Units of the instruments is under investigation.

3.3 Array Elements Manager Specification Requirements

This section describes the requirements categories needed for the software modules of the AECS which interface with ACADA, named Manager (see [AD-1] for details).

3.3.1 Interfaces with ACADA

The control interface between ACADA and the AE Managers are specified in the following documents:

CTA-ICD-SEI-000000-0002 ICD for ACADA - Generic Telescope Control
CTA-ICD-SEI-000000-0003 ACADA - FRAM Control ICD
XXX ACADA - LIDAR Control ICD (TBD)
XXX ACADA - FRAM Control ICD (TBD)
XXX ACADA - All Sky Camera (ASC) Control ICD (TBD)
XXX ACADA - Illuminator Control ICD (TBD)

3.3.2 State Machine

In describing the process of developing Manager Requirements for each Array Elements, the *Deterministic Finite State Machine* approach is mandatory, as described in [AD-1], and the states and modes defined must be compliant with the State Machine required to be controlled by ACADA:

- Generic Telescope State Machine [AD-7]
- LIDAR State Machine [TBD]
- FRAM State Machine [TBD]
- Weather Station State Machine [TBD]
- ASC State Machine [TBD]
- Illuminator State Machine [TBD]

3.4 Monitoring, Storage and Display Specification Requirements

The monitoring, storage, and display of the non-safety related parameters of all the controllable items, together with the storage service for the IPS parameters, will be performed though the monitoring and logging system of ACADA, when possible [AD-1].

The communication protocols are specified in Section 3.2.

The monitoring interface between ACADA and the controllable items are specified in the following documents:

- Array Element Monitoring ICD [AD-13]
- Array Element Logging ICD [AD-14]
- PMS Monitoring and Logging ICDs [TBD]
- ICT Monitoring and Logging ICDs [TBD]
- IPS Logging ICDs [TBD]

Note: The compatibility of the technical features of the controllable items with the monitoring/storage technology adopted for ACADA is under investigation. If some incompatibility with ACADA will be found, more specific instruction will be provided.

4 Sub-systems Definition for Array Elements Control Systems

This Chapter provides inputs to develop the hardware and software component of the sub-system related to the Array Elements Control Systems (AECS), the most complex customized control systems, which includes the LCS and the Managers as defined in [AD-1]. Anyhow the same approach can be applied to every CTAO control system which need customized solutions for the realization or when an integration of different industrial solutions needs to be defined.

4.1 Design Specification

This section describes how to define the design specifications needed for the safety and non-safety control systems to be developed for the Array Elements. The content of this section applies to the Preliminary Design and Detailed design stages of the development cycle.

Verification Plan

Define a Verification Plan [AD-6] considering the following aspects:

- Formulation of the safety and non-safety criteria and states.
- Test patterns/Test Cases for the inputs and its environment.
- Procedure for validation of each safety and non-safety function.

In the process of defining Verification activities the operation and maintenance processes should be taken into account.

4.1.1 Hardware Design

The hardware design specification of safety and non-safety control systems, after the preliminary design and the detailed design stages, should contain the following items:

- Generation of block and circuits diagrams
- Interface criteria
- Design analysis
- Detailed calculations and test procedures.
- Arrangement drawings
- Assembly drawings
- Detail drawings
- Installation drawings
- Logic diagrams
- Numerical control drawings
- Schematic diagrams
- Models and simulations

4.1.2 Software Design

Software design documentation should contain safety-related functions and non-safety-related functions. The software items architecture, design requirements, implementation logic, and data structures must be documented. In particular the detailed design stage should produce design specification at module level for each software that need to be implemented (the firmware of the field devices selected as industrial solution are not included in this procedure), which are usually LCS Supervisors (e.g. PLCs) and the Managers:

- Software Module Design Specification
- Module test specification.

The module design will enable the implementation of software, which not only achieves the required integrity level, but which is also analyzable, verifiable, and maintainable.

The Software Module Design Specification should indicate how independence between safety-related and non-safety related function is achieved with the objective to create the software of a defined integrity level from the software design specification.

The software produced should be minimum in size and complexity, and the module design method chosen should possess features that facilitate:

- The abstraction, modularity, and other features which control complexity.
- The clear and precise expression of functionality, information flow between components, sequencing and time related information, concurrence and data structures and properties.
- Human comprehension.
- Verification and Validation.

The module design should include self-monitoring of control flow and data movements and On failure detection appropriate actions should be taken (see next Section).

If standard or previously developed software is to be used as part of the design, then it should be clearly identified and documented.

Wherever possible existing verified software modules (or function blocks) should be used in the design.

Each software module should be readable, understandable, and testable.

4.1.3 Fault Management

The fault management is one of the most important features of the software design and must be applied at all level of control for the correct operation of the system.

The software associated to every level of the AECS (LCS and Manager) should independently be able to identify, isolate and mitigate with the appropriate actions the fault situations related to the hardware and software that can occur in the associated sub-system.

The following classes of potential faults should be considered to identify distinct failures that can arise and to give them a characterization:

- Failure of some integrity property of the state of the system. This may either be an inconsistency arising from the internal variables of the intended software or, in the input/output relation computed by the program.
- Failure to meet some progress or timing constraint. This will translate to a requirement that expresses that the software delivers certain responsiveness.
- Failure to obtain the format/range etc. of data from some hardware interface.
- Failure of the system software, kernel failure (whenever possible).
- Failure of compiler/assembler/translator from the coding language to the executed code.

The measures to control failures are built-in features of the safety and non-safety control systems, while the measures to avoid failures are the procedures done during the development life cycle.

It is important to have in mind that it is not possible to list all the individual physical causes of failures in complex hardware and software systems (e.g. Random failures in the hardware, failures caused by incorrect use, software failures).

To avoid such failures or control such failures when they occur many measures are necessary. In the Control System Requirements Specification an appropriate group of measures and techniques to be used for safety integrity, as they relate to the control of failures and the avoidance of failures, should be composed.

4.1.4 BIT tests

As stated in [AD-1], to support operational and maintenance of the CTAO control systems, techniques such Built in Test (BIT) and the related equipment, should be designed and developed whenever is possible and reasonable considering the available budget and manpower efforts, providing very good solutions for self-diagnosis and self-recovery mechanisms. See Table A.5 for more details.

4.1.5 Safety Units and Safety Functions

The safety functions and the related Safety Unit are intended as defined in [AD-1].

The safety functions are required to put the equipment under control (EUC) into a safe state or to maintain a safe state, as intended in [AD-1].

The design specification of the safety functions, as well as the Safety Units of the Array Elements, must follow the guidelines provided by the related standard, as stated in [AD-2].

In defining the Safety Functions the correct Safety integrity level (SIL) must be considered, as a measurement of performance required for a safety instrumented function (SIF), as specified in [AD-2].

Based on the nature of the EUC, the Safety Units should be designed as separate dedicated system, or they must be integrated with the normal machine control system:

- The hardware and software dedicated to performing a particular safety function, like limit switches intervention or emergency stops, must be considered as stand-alone equipment.
- Functions available in the machine control system, like Safe Torque Off (STO) or Safe Stop 1(SS1) of motor drive, must be integrated in the other safety equipment.

An important aspect of the design of safety equipment is that the system must continue to operate correctly under all foreseeable conditions, to provide the safety functions. A failure of the safety functions can result in an immediate increase of the risks of using the equipment. The main task of the safety system designer is to prevent hazardous conditions and to prevent unexpected start-up and to avoid damages to the mechanical components.

Relevant means of implementing safety functions include electro-mechanical relays (e.g., Interlocks), non-programmable and programmable solid-state electronics. Programmable electronic safety-related systems typically incorporate programmable controllers (Safety PLCs), programmable logic controllers (Safety Logic Chains), microprocessors, application specific integrated circuits, or other programmable devices adapted for safety (for example "smart" devices such as sensors/transmitters/actuators) as specified in [AD-2].

4.2 Implementation

This section describes how to tailor the implementation of the control system components to be developed for the Array Elements. The content of this section applies to the Implementation stage of the development cycle.

4.2.1 Hardware

The hardware for the different sub-systems of the Array Elements is procured and the following two stages may be applied:

- *Prototyping/Production Engineering.* This is the prototype stage of the development. Prototypes are tested prior to the continuation of the development. Any prototype software that may be required is used at this stage for testing.
- *Sub-system Integration and Proving*. This is the stage at which the hardware documentation is collated, showing test results and final production information - prior to system integration.

4.2.2 Software

The software implementation should be done following the *Software Module Design Specification* defined during the design stages. The activity of the implementation is the *Coding* which produces the *Source Code* and the supporting documentation. The objective is to implement software, which achieves the required integrity level, and which is also analyzable, verifiable, and maintainable.

The developers must follow the standards provided by CTAO for the software implementation of each level of the control system [AD-2] and briefly summarized below.

Sub-System	Language		Style
LCS Supervisor	PLC (IEC 61131-3)	Instruction List (IL) Structured Text (ST) Ladder Diagram (LD) Function Block Diagram (FBD) Sequential Function Chart (SFC).	-Small scale applications typically are programmed by the user -Large scale applications (e.g. in control) should be configured from standard function blocks as completely as possible. -Only those functions for which no standard block is available, should be programmed. This minimizes coding errors.
Safety Logic	Safety PLC (IEC 61131-3)	Certified Safety Function Blocks	Functional Safety [AD-2]
	Certified Functional Safety tools and languages		Functional Safety [AD-2]
Managers	Software Programming Standards [AD-3]	Java, Python, C++ (using ACS framework)	Software Programming Standards [AD-3]

Table 5: Language and Style. The reference to [AD3] is related to the languages to be used.

The language(s) chosen should, at least, meet the following requirements:

- A programming language should be selected that relates to the characteristics of the application.
- The language chosen should contain features that facilitate the identification of programming errors.
- The language chosen should support features that match the design method.

As a minimum, the following information should be contained in the source code documentation:

- Version
- Author
- Description
- inputs and outputs
- configuration history.

4.2.3 Array Managers

The Array Element Managers, listed in [AD-1], should be implemented as ACS Characteristic Components. See more information in [AD-8].

4.2.3.1 Realization of interface with ACADA

The Array Element Managers expose their software interfaces to ACADA using the ACS communication mechanisms. In particular, control operations, and monitor and control points are realized as CORBA Interface Definition Language (IDL) files. The IDL file creation is driven by the ACADA team, and its content and any change must be agreed upon by both the ACADA and AE team plus IM, and put under configuration control on the CTAO gitlab repository. The interface exposed as IDL files must be

compliant with the corresponding ICD. Any non-trivial change on the released IDL files must happen as a consequence of a Change request accepted by the CTAO CCB.

The currently existing IDL are locate in [AD-9].

4.2.3.2 Code implementation

Array element managers code can be implemented using any of the ACS-supported languages: C++, Java and Python. Examples and tutorials are provided in [AD-10].
The code must be maintained under configuration control in the CTAO GitLab repository.
The AE Manager code must follow the standard software QA practices and the CTAO programming standards [AD-3]. In particular, the code must be provided with unit and integration tests, with a minimum of 80% (TBD) line coverage.

Some example implementations are provided here [AD-11].
The Code must be properly tested and tagged before installing it at the On-site ICT environment (details TBD).

4.2.3.3 Runtime environment

In production, the AE Managers run on dedicated nodes of the Onsite-ICT infrastructure.
Onsite-ICT machines use Linux Red Hat distribution derivative operating system.
The usage of docker containers is permitted (details TBD).

4.3 Assembly, Integration and Test Stage

This section provides some inputs to perform the integration and testing stage of the control systems to be developed for the Array Elements. The content of this section applies to the AIT stage of the development cycle.

The prerequisites are:

- The Hardware of the Subsystem is selected for every control level (LCS and Managers)
- The application software developed for every control level is verified (Managers)

The objective of the AIT activity is to integrate the verified application software with the hardware (e.g. Target PLC) and develop confidence that the system will meet the specification and Requirements.

The integration and testing of the AECS should be performed starting from the lower level up to the highest level:

- Integrate and test the verified LCS (hardware and software). The verification and integration of supervisors and filed devices (e.g. actuators, motors) is considered included in the verified LCS.
- Integrate and test the verified Supervisors (hardware and software) with the verified Managers.

4.3.1 Integration

As part of the integration activities, evidence needs to be collected showing that system functionalities and safety (non-safety) requirements will be maintained during operation.

To achieve this, the following should be investigated:

- The effects of failure of individual software and hardware modules on the maintenance of system requirements.
- The effects of a single control level failure (e.g. PLC supervisor) on other levels and the effects of inter- level communication failures.
- Safety aspects of software and hardware system architecture and their behavior in the event of operation e.g. fault tolerance through redundancy, exception handlers, recovery blocks etc.

4.3.1.1 Mock-up versions

In order to facilitate the off-line testing of the AE itself and pre-integration with ACADA, mock-up versions of the AE managers, which do not require any AE instrumentation to run and can run in any standard Linux Red Hat derivative distribution currently supported versions defined at [TBD reference] derivative machine must be delivered.

4.3.1.2 Integration with ACADA

In order to integrate, AE managers and ACADA use the same ACS manager instance at runtime. The AE and ACADA team will work together to define and realize integration tests based on the ACADA use cases. In addition, they will work together to define common software settings (ACS CDB) for concurrent operations of the respective systems. Such settings are to be maintained under configuration control after the AE acceptance.

4.3.1.3 Unit Testing

As part of code implementation and before the Integration with ACADA, the developers of the AE Managers are required to implement and perform Unit Tests of their code (in addition to the functional tests), by using appropriate tools for the language used. A list of tools is available in 2.7.1.

4.3.2 Testing

After the integration and installation of the hardware and software part of the safety and non-safety related systems, a testing of the safety and non-safety functions implemented is mandatory.

This phase involves the presence of the following prerequisites:

- Specification of the safety (non-safety) functions.
- Outcome of prior off-line-tests and type approvals.
- Checkout procedure for the correctness of the installation.
- Start-up and shut down procedures.
- Run in procedures (if appropriate for controlled process).
- Manually executed complementary safety shut down procedure (if appropriate for controlled process).

The activities are:

- Testing the correctness of the installation.
- Running in the controlled process (if applicable).

- Testing of all automatic safety related functions during run-in.
- Start-up of the controlled process.
- Testing the safety functions with manual safety functions and with simulated dangerous conditions.
- Shut down of the controlled process.

The main goal will be the testing of the System - installation, pipes, vessels etc. - and of the basic process control functions - sensors, actors, Human-Machine Interface, operating procedures - as well as the operators gaining experience. This time is when all the safety functions are tested in the integrated environment and monitor the process response.

4.3.3 Safety-Functions Testing

The testing of some safety functions requires the process to be in a dangerous state. The test may demonstrate that the safety system would bring the process into a non-dangerous state by the appropriate action (e.g. shut down). In test steps like these, any "real danger" must be avoided. This can often be done by taking these test steps in the run-in phase, or, by having complementary (manual) shut down facilities available.

An alternative way is to simulate a dangerous state of the process, at the sensor level, to test some of the safety functions.

The testing of the safety functions must be planned and carried out very carefully. On one hand the testing must be completed in a specific sense (cover all functions) - on the other hand the on-line-testing of safety functions can be expensive and potentially dangerous.

It has to be noted that an important key to a later safe operation of the process is the involvement of System operations, and maintenance personnel, during the on-line-testing phase to assure a clear understanding of all aspects of the process, the control system, and the safety-related system.

5 Verification and Quality Assessment

In this chapter are presented some guidelines on how to apply the Verification and Quality activities for the development process of the CTAO customized control systems (e.g. AECS) and the integration of industrial devices. Whenever needed, specific instructions for the AECS will be provided.

The full procedure, roles and responsibilities of the Verification and Quality activities must be defined to be compliant to the guidelines and procedures defined at CTAO level [AD-6]. The Validation process, which provides the organization of external review, is outside the scope of this document.

This section reports the assessment that usually a quality/verification process should achieve for every AECS and to which every software and hardware component of the control system must undergo (LCS, Managers). The activities are performed through the development cycle (see Section 2).

5.1 Audits and Review during the Cycle

During the development phase of the AECS, the teams should conduct the Verification and Quality assessment, by organizing internal reviews (e.g., system, subsystem, component, life cycle processes, test readiness, production approval) and audits (e.g., functional and design configuration), for the purpose of assessing technical progress together with the quality of the product and process.

The process should have 'hold points' beyond which development should not proceed until the review/audit assess the compliance with the deliverables and quality levels required in that specific stage.

Reviews may result in the need to iterate through the Iterative process to resolve identified deficiencies before progressing further in the development activity.

The purpose of the Verification and Quality activities is to ensure that the control system is being implemented into the product with the desired quality as the development proceeds, so that if problems occur, they can be resolved as soon as they manifest themselves.

Each team must establish and maintain product and process quality factors to continuously improve products and processes throughout the system life cycle in a manner consistent with CTAO objectives and Quality Requirements [AD-6]. General Verification and Quality activities to be performed for each development stage are reported in Table 6.

System Definition	
Quality Activity	**Verification Activity**
Definition of quality factors at system level: Producibility, Verifiability, Ease of distribution (packaging, handling, transportation, storage, installation, and transition), Usability, Supportability, Trainability, Disposability.	Perform reviews to assess the maturity of the system development effort and the readiness to progress to subsystem definition.
Preliminary Design	
Quality Activity	**Verification Activity**
Identify and quantify quality factors defined above for sub-systems. The subsystem life cycle quality factors should be decomposed and allocated among the assemblies, and then the components, in a manner that ensures that quality-factor traceability is maintained.	Perform preliminary design review to: • Assess the maturity of the definition of the AECS and the related sub-systems. • Determine whether the total system approach to detailed design satisfies the system baseline • Unacceptable risks are mitigated • Issues for all subsystems, products, and life cycle processes are resolved.
Detailed Design	
Quality Activity	**Verification Activity**
Identify and quantify component life cycle quality factors, which influence each component's capability to meet downstream requirements. The component life cycle quality factors should be decomposed and allocated among the component's subcomponents, and then lower subcomponents, in a manner that ensures that quality-factors traceability is maintained.	A detailed design review should be planned to assess the maturity of the development effort and to check whether the design of the AECS is ready to continue into AIT stage. The related technical reports must be provided.

Table 6: AECS Verification and Quality Activities.

5.2 System Verification Testing during the cycle

The verification testing of the system is applied is usually applied during the AIT and Verification stages to test the integrated system to ensure compliance with the Requirements Specification of the overall process in which all the sub-systems (Field Devices, Supervisors and Managers) of the AECS has been integrated and tested. Testing should be the main verification activity, but simulation and modelling may be used to supplement the verification process.

Some examples are provided in Appendix.

For the overall verification the following tests, compliant with the system requirements, should be performed at system level:

- Function tests: all sequences, all functions, all inputs, all outputs.
- Time behavior tests: all operating conditions, all timing requirements.
- Interface tests: all interfaces, all operational modes.
-

The Verification should identify:

- All hardware and software used.
- All the equipment used.
- All the equipment calibration.
- All simulation models used.
- All discrepancies found.

'Test' is used here in its wider sense, as encompassing all kinds of verification process. All information related to the assurance work on AECS for critical applications should include the characteristics and aspects summarized below.

5.2.1 Presentation of the test objective and tools.

This part should include an overview of the test object based on the requirements specification and functional analyses. The following information should be derived:

- Presentation of the test object
    - Purpose, application, function.
    - General data and specification.
    - Safety and non-safety concept of the whole system (hardware and software).
    - Explanation of the structure and function of the hardware.
    - Explanation of the structure and function of the software.
- Presentation of the test tools
    - Applied test methods.
    - Applied test equipment.

5.2.2 Detailed description of the tests.

In this part the test carried out should be listed and their results should be described in sufficient depth to ensure complete understanding. The description part consists of all measurement configuration, tables and diagrams (e.g. fault trees, FMEA-tables, flow-charts, listing reviews, specific test results, wiring diagrams etc.) made in course of the whole testing work.

Tests that are available include:

- For hardware
  - Functional analyses.
  - Detailed failure analyses (FMEA, Error Sequence Analysis, quantification etc.).
- For software
  - Functional analyses (black-box, white-box, etc.).
  - Detailed failure analyses (investigations regarding structure and dependence/ independence of software modules).

5.2.3 Summary and Final Evaluation.

This should provide a short description of the whole test work and it should include:

- Functional principles of the test object
- Tests carried out
- Test conditions
- Applied test methods
- Final judgement in relation to the requirements.

5.2.4 AE Managers Verification and Quality

The AE Managers must guarantee an appropriate quality level and a proper verification coverage, at least for the implementation part, to guarantee an efficient integration and operation with ACADA.

5.2.5 Software Maintenance Provisions

Any revision on AE managers code to be installed on the array needs to be properly tested and tagged. Updating on the AE code must be coordinated with the TBD stakeholders. After any update a full regression test must be carried out.

Appendix A– Iterative Process Tasks and Activities

Requirements Analysis and Validation

The Requirement Analysis stage of the Iterative Process aims to determine the needs or conditions to meet for the control systems under development, for purpose of establishing:

- what the system will be capable of accomplishing;
- how well system products are to perform in quantitative, measurable terms;
- the environments in which system products operate;
- the requirements of the human/system interfaces;

- the physical characteristics;
- the constraints that affect design solutions.

The tasks and activities associated with requirements analysis are described Table A.1. For more details see [4].

After the analysis of requirements, the teams should perform the tasks of requirements validation (summarized in Table A.2 of the Appendix) to evaluate if:

- the established requirements baseline represents the identified stakeholder expectations and project constraints
- the system life cycle concepts have been adequately addressed.

Tasks	Activity
Define and quantify the Stakeholder expectation	- Define Functional Requirements [life cycle processes, and desired quality factors] - Define Performance Requirements [how well each function is to be accomplished] - Constraints [funding; cost or price objectives; schedule; technology; design characteristics; hours of operation per day; on-off sequences; external interfaces; and specified existing equipment, or procedures related to life cycle processes]
Identify and define constraints that impact design solutions	-Approved specifications and baselines developed by CTAO -Engineering and technical plans/domain technologies -Team assignments and structure/ Policies and procedures -Automated tools availability -Required metrics for measuring technical progress -Reuse and commercial-off-the-shelf (COTS) -Physical, financial, and human resource allocations to the technical effort
Identify and define operational scenarios for use and application of the control system	- Define expected interactions with the user and other systems - Define physical interconnections with interfacing systems, platforms, or products
Defines system effectiveness measures (MOE)	-Reflect overall stakeholder expectations and satisfaction, including performance, safety, operability, usability, reliability, maintainability, time and cost to train, workload, human performance requirements
Define System boundaries	-Which system elements are under design control of the team and which fall outside their control -The expected interactions among system elements under design control and external and/or higher level and interacting systems outside the system boundary
Define External interfaces	-functional and design external interfaces - Identify interacting high-level systems (mechanical, electrical, thermal, data, communication-procedural, human-machine)
Define utilization environment for each operational scenario	-Identify factors for system minimization of the potential for human or machine errors or failures that cause injurious accidents or death. - weather conditions (e.g., rain, snow, sun, wind, ice, dust, and fog), temperature ranges, topologies (e.g., mountains, deserts), biological (e.g., animal, insects, birds, and fungi), time (e.g., day, night, and dusk), induced (e.g., vibration, electromagnetic, acoustic, and chemical)
Define life cycle process concept	Analyze outputs of previous tasks to define life cycle process requirements necessary to develop, produce, test, distribute, operate, support, train, and dispose of system products under development (Manpower, Human Engineering and Safety must be included)
Define Functional and Performance Requirements	- Perform functional context to define what the system should be able to do - Define the performance requirements for each function of the system
Define Modes of Operation	Defines the various modes of operation (embedded training capability, fully operational, etc.) and the conditions (environmental, configuration, operational, etc.), which determine the modes of operation
Define Technical Performance measures	Identify the technical performance measures (TPMs), which are key indicators of system Performance.

Table A.1: Tasks and Activity of the Requirements Analysis stage of Iterative Process

Tasks	Activity
Compare to stakeholders expectation	- analyzes and compares the established requirements against stakeholder expectations - ensure that the technical requirements adequately represent the stakeholder's needs, requirements, and constraints
Compare to project constraints	Ensure that the technical requirements correctly represent, and stay within, enterprise and project policies and procedures, acceptable risk levels, plans, resources, technology limitations, objectives, decisions, standards, or other documented constraints.
Compare to external constraints	To ensure that the specified technical requirements correctly represent, and stay within, applicable national and international laws (including environmental protection, hazardous material exclusion lists, waste handling, and social responsibility laws); correctly state external interface requirements with existing or evolving systems, platforms, or products; include applicable general specification and standard provisions affecting the development; and adequately define competitive product capabilities and characteristics.
Identify variances and conflicts	The project identifies and defines variances and conflicts that arise out of the validation tasks. Each variance or conflict is resolved by iterating through requirements analysis to refine the requirements baseline.
Establish validated requirements baseline	Once the established requirements baseline variations and conflicts are satisfactorily resolved, the requirements baseline is considered valid. This validated requirement baseline is then used as input to functional analysis (see 3.3) and documented in the integrated repository.

Table A.2: Tasks and Activity of the Requirements Validation stage of Iterative Process

Functional Analysis and Verification

The goal of functional analysis stage is to accomplish the following objectives:

- describe the problem defined by requirements analysis in clearer detail
- decompose the system functions to lower-level functions that should be satisfied by elements of the system design (e.g., subsystems, components, or parts).

This is accomplished by translating the validated requirements baseline into a functional architecture, which describes the functional arrangements and sequencing of subfunctions resulting from decomposing the set of system functions to their subfunctions.

The project should conduct the functional verification to assess the completeness of the functional architecture in satisfying the validated requirements baseline and to produce a verified functional architecture for input to synthesis. The tasks associated with functional analysis and Verification are identified in Table A.3 and A.4. For more details see [4].

Tasks	Activity
Functional context and functional behaviour analysis	- Analyze each system function to determine the responses (output) of the system to inputs necessary to accomplish system objectives - Understand the functional behavior of the system under various conditions and assess the integrity of the functional architecture
Define functional interfaces	Defines functional interactions and identify interfaces
Allocate performance requirements	Performance requirements are divided into allocable sets and are directly allocated to functions
Functional decomposition	Decompose the system into subfunctions based on what the system must accomplish. Risk analyses are performed to select a balanced set of subfunctions and to allocate performance requirements to subfunctions to assure that the functional architecture satisfies the system requirements.
Define subfunctions	Functions are decomposed in terms of their functional behaviors, states and modes of operation, functional time lines, conditions for control of data flow, functional failure modes and effects, and potential hazard monitoring functions that are needed

Establish functional architecture	Establish the functional architecture, appropriate to the level of development, to define the allocation of performance requirements from which design solutions should be determined via synthesis.

Table A.3: Tasks and Activity of the Functional Analysis stage of Iterative Process

Tasks	Activity
Define verification procedures	Defines the procedures for verifying the established functional architecture
Conduct verification evaluation	Conducts defined procedures to: -Verify architecture completeness -Verify functional and performance measures -Verify satisfaction of constraints
Identify variances and conflicts	-If non-required functions and/or performance requirements were introduced during functional analysis then functional analysis tasks are repeated to correct voids and to eliminate non-required functions and/or performance requirements. -If valid functional and/or performance requirements were derived they need to be reflected in the requirements baseline and requirements analysis and validation should be repeated to produce a revised, validated requirements baseline
Establish verified functional architecture	The functional architecture is verified upon satisfactorily resolving the variances and conflicts identified.

Table A.4: Tasks and Activity of the Functional Verification stage of Iterative Process

Synthesis and Design Verification

The synthesis activity is performed to define design solutions and subsystems requirements based on the verified functional architecture. It translates the functional architecture into a design architecture. The most important tasks associated with synthesis are reported in Table A.5 (for more details see [4]) and involve selecting a preferred solution from a set of alternatives and understanding associated cost, schedule, performance, and risk implications.

The established design architecture, appropriate to the level of development, should document the design solution and interfaces. The design architecture includes the requirements traceability and allocation matrices, which capture the allocation of functional and performance requirements among the system elements.

The Verification process of the design architecture produced should be accomplished to demonstrate that the architecture satisfies both the validated requirements baseline and the verified functional architecture.

The tasks associated with design verification stage are Summarized in Table A.6.

Tasks	Activity
Group and allocate functions	Group common functions and subfunctions of the verified functional architecture into logical functional elements in a manner that permits their allocation to design elements.
Identify design solution alternatives	Generate alternative design solutions for the functional elements identified (hardware, software, material, data, facility, people, and techniques).
Assess different design alternatives	Analyzes alternatives to: - Identify potential hazards to the system, humans involved in the system and supporting the system life cycle processes. - Determine the degree to which quality factors (producibility, testability, ease of distribution, usability, supportability and disposability) have been included in the solutions. - Identify the design characteristics and human-engineering elements associated with life cycle quality factors.

	- Determine the technological needs necessary to make the design solution effective. - Identify and define the physical interfaces among products, subsystems, humans, life cycle processes, and external interfaces to higher-level systems or interacting systems. - Assess whether use of standardized end items would be technologically and economically feasible. - Determine availability of an off-the-shelf item (non-developmental hardware or software) - Assesses failure modes, the effects, and the criticality of failure for design alternatives. - Assesses the testability of design alternatives to determine built-in test (BIT) and/or fault isolation test requirements to support operational or maintenance considerations.
Identify make-or-buy alternatives	Perform economic analysis of design alternatives to support make-or-buy decisions to address whether it is more cost-effective for the project to produce the design element vs. going to an established supplier.
Design selection and verification	Develops models and/or prototypes to assist in: -Verify that the selected design solution (made up of hardware, software, material, humans, facilities, techniques, data, and/or service) meets allocated functional and performance requirements, interface requirements, workload limitations, and constraints. -Verify that the selected design solution satisfies functional architecture and requirements baseline requirements.
Assess FMEA	A failure modes and effects analysis (FMEA) should be used to identify the strengths and weaknesses of the selected design solution.
Assess testability needs	BIT mechanisms should be provided for the elements that are normally maintained by the operators, users, or field support engineers belonging to the selected design solution. BIT can be used for diagnostic operations to support lower-level maintenance actions.
Finalize design	Finalize the design for the selected alternative. The designation and description of interfaces (internal and external) among design elements are finalized.
Produce integrated data package	Completes the drawing, schematics, software documentation, manual procedures, etc., as necessary, to document the selected design elements in an integrated data package.
Establish design architecture	Establishes the design architecture, appropriate to the level of development, to document the design solution and interfaces. The design architecture includes the requirements traceability and allocation matrices, which capture the allocation of functional and performance requirements among the system elements. Design architecture definitions should be documented in the integrated repository, along with trade-off analysis results, design rationale, and key decisions to provide traceability of requirements up and down the architecture.

Table A.5: Tasks and Activity of the Synthesis stage of Iterative Process

Tasks	Activity
Define verification activity	- Select Appropriate Verification Method (Inspection, Analysis, Demonstration, tests) - Develop a verification Matrix and select the models or prototype to be used - Define procedures for each verification methods selected and defines the criteria for determining the success or failure of the procedure for planned and abnormal conditions
Conduct verification evaluation	The verification activity should verify the following: - Design elements descriptions are traceable to functional architecture requirements - Functional architecture requirements are allocated to the design architecture - Internal and external design interfaces are traceable to their source requirement - Evaluation results satisfy the constraints of the functional architecture - Constraints of the established design architecture are traceable to the validated requirements baseline
Identify variances and conflicts resulting from verifying activities	When variances show incompleteness, synthesis tasks or functional analyses tasks are repeated to correct omissions: - to eliminate non-required functions and/or performance requirements - to produce a new validated requirements baseline and verified functional architecture.
Verified design architecture	The design architecture is verified upon satisfactorily resolving the variances and conflicts identified and documented in the integrated repository.
Establish specifications and configuration baselines	Develop/update product and interface specifications appropriate to the stage of development for each element of the design architecture. Develop/update appropriate configuration baselines for each element of the design architecture. The hierarchy of specifications (product and interface) for the design architecture forms the specification tree appropriate for the stage of development.

Develop system breakdown structure	The project develops an SBS for the system designed, including life cycle process requirements

Table A.6: Tasks and Activity of the Design Verification stage of Iterative Process

System Analysis and Control

Systems analysis provides a rigorous quantitative basis for establishing a balanced set of requirements and for ending up with a balanced design. The most important tasks associated with systems analysis are identified in Table A.7.

The control activities are applied for the purpose of managing and documenting the activities of

the Iterative Process. The tasks associated with control are identified in Table A.8 and provide the following:

- A complete and up-to-date picture of Iterative Process activities and results, which are used in accomplishing other activities
- Planning for and inputs to future applications of the Iterative Process
- Information for production, test, and support
- Information for decision makers at technical and project reviews

The Control activities are intended to be practiced within internal team only and must be organized based on the central guidelines provided by the CTAO for the related tasks [Ref].

Tasks	Activity
Assess requirement conflicts	Assesses conflicts among requirements and constraints identified during requirements analysis to identify alternative functional and performance requirements, where necessary.
Assess functional alternatives	Assesses possible alternative subfunction arrangements for the decomposition of a function and for the allocation of allocable performance requirements to the subfunctions during functional analysis
Assess design alternatives	Assesses potential groupings and allocations of functions from the verified functional architecture and identified design alternatives during synthesis
Identify risk factors	Assesses and constraints requirements made during synthesis, and design elements of the design architecture to identify the risk factors.
Quantify risk factors	Quantify the impact of identified risk factors on the system. For system effectiveness assessments, each element of the system architecture is assessed to determine what can go wrong, and if it goes wrong, what impact it may have on the system.
Design effectiveness assessment	Determine the system design effectiveness based on the results of the assessments and analyses. The results of the assessments and analyses are documented in the integrated repository and briefed at appropriate technical and project reviews.

Table A.7: Tasks and Activity of the System Analysis stage of Iterative Process

Tasks	Activity
Technical management	Manage the tasks and activities of the Iterative Process to control data generated, configuration of the design solutions, interfaces, risks, and technical progress. Setting up appropriate repositories and procedures is included.
Track systems analysis and test data	Collect, analyze, and tracks data from systems analyses to document activities, rationale, recommendations, and impacts, and from tests to document results, variances, and follow-up activities.
Track requirement and design changes	Collects and sorts data to track requirement and design changes and to maintain traceability of change source, processing, and approval.
Track progress against project plans	Collects and sorts data reflecting plan activities and tracks progress against the engineering plan, master schedule, and detail schedule.

Track product and process metrics	The project collects, analyzes, and tracks product and process metrics to: - Determine technical areas requiring project management attention - Overall system quality - Early detection of problems
Update specifications and configuration baselines	Update specifications and configuration baselines to reflect all changes.
Update requirements views and architectures	Update requirements views and the functional, design, and system architectures to reflect changes brought about by an acquirer, systems analysis, validation and verification deviation, or management decision.
Update technical plans	Update the technical plans to reflect changes brought about by an acquirer, systems analysis, plan activity deviation, or management decision.
Integrated repository	Establish and Maintain a repository of all pertinent data and information from previous tasks.

Table A.8: Tasks and Activity of the Control stage of Iterative Process.

Appendix B– Verification Activity

PLC Verification Procedure

Since the PLC technology is wide used for the development of the low-level Array Element Control Systems (LCS) here it is presented a verification method that usually is applied to the PLC-based systems. This method normally concentrates on the application software and the integration of the whole system to demonstrated that the hardware as well as the software system meets the requirements for a specific application. The procedure is divided into the **Analytical Approval** and **Operational Approval.**

The following concrete steps are conducted during the **Analytical Approval**:

- Inspection of the documentation with respect to completeness, validity and consistency
- Verification and quality evaluation of logic diagrams
- Verification and quality evaluation of the application software:
    - The application software is verified with respect to the specifications and the logic diagrams which have to be pre-verified by the process engineer.
    - Extraction of all safety critical parts (modules, subroutines) of the software.
    - Control flow and data flow analysis.
    - Verification of all specified functions.
    - If possible, simulation of all functions and time behavior under normal and erroneous conditions on a simulator (recommended).
    - Verification of hardware design and installation documents.

The following stages are conducted during the Verification/Quality process during the **Operational Approval**.

- Test of the installation of the PLC system with respect to:
    - Field wiring (e.g. separate installation of redundant wiring).
    - Protective and functional earthing.
    - Noise and transient suppression measures (separation of cables for inputs, outputs and power circuits, correct length of wiring, separation of the field wiring from internal I/O cabling and from bus lines, control of mechanical contacts which are in series with inductive loads).

  - Test of compliance with the actual service and environmental conditions (e.g. temperature, shock and vibration, electromagnetic influence).
  - Interaction between PLC and process periphery (loop checks)
  - Binary I/O, checking binary and digital input signals to ensure that physical states of sensors comply with signal latches (memory elements) in the PLC, checking that no forced binary and digital outputs are set.
  - Analog I/O, checking analog input signals to ensure equivalence of physical value and data received by the PLC. Supervised inputs and outputs: detection of opens and shorts.
- Test all system functions. During these tests, the functional behaviour of the system with respect to the specified functions is validated. All functions are being tested under operational conditions.
- Perform Fault simulations that are conducted based on a pre-defined list of failures, which involves:
  - Sensors, contacts and actuators.
  - Inputs and outputs.
  - Field wiring (e.g. exchanged connections).
  - Interlocks.

  During failure simulations, it may be validated that the system (in the case of failures) is being brought into a safe state.