IT Architecture Principles and Implementation Guidelines test

Copernicus Land Monitoring Service

Author: **European Environment Agency (EEA)**

Date: **2025-03-06**

Version: **1.4a**

Index

1. Preface

The EEA (European Environment Agency) CLMS IT architecture principles are indicative and must be evaluated in all IT deliverables.

2. Introduction

The IT architecture principles set the overall framework for the EEA CLMS IT landscape. The principles are designed to ensure a consistency in deliverables and at the same time support the CLMS program's IT vision and -strategy. The principles are designed to ensure that IT solutions are coherent, can be further developed and operated efficiently, that they support business needs and security requirements, etc.

A uniform approach is required to ensure the coherency goal. The EEA CLMS programs IT applications may depend on and interact with each other. It is therefore important that IT solutions focus on connectivity and potential synergy effects to ensure continued coherence in the IT landscape.

Any application provided may be developed, operated, maintained, and further developed by a supplier different from the supplier who delivered the initial application. Therefore, efforts must always be made to be supplier independent. Other suppliers must be able to continue working from where the previous supplier left off.

3. Scope and key terms

The scope of the EEA CLMS IT architecture principles is IT solutions to be delivered to the CLMS. The solutions delivered will include functionalities required to support the program (for Example, maintain and operate CLC+ Core multi-use grid-based Land Cover/Land Use hybrid data repository). This includes also the dependencies of these IT solutions to other internal or external systems.

Definition of the key terms used within this document:

Application Programming Interface (API) - is a set of protocols, tools, and definitions that allow different software applications to communicate with each other. It defines the methods and data structures that developers can use to interact with a service or application, facilitating the exchange of data and functionality.

Automation scripts - refers to sets of instructions, written in scripting languages designed to automate repetitive tasks and processes. These scripts streamline workflows, reduce the need for manual intervention and ensure consistency and efficiency in performing tasks.

Client specific software/IT solution - is a custom-designed software/solution that is tailored to meet the unique needs, requirements, and preferences of a particular client or organization.

Commercial software - software products that are developed, marketed, and sold for profit by software companies or developers. Commercial software is typically licensed to end-users, who must purchase it or pay a subscription fee.

Continuous Integration and Continuous Deployment (CI/CD) - are closely related methodologies designed to streamline and automate software development. Together, they ensure that code changes are continuously tested, integrated, and deployed to production environments, enabling teams to deliver updates more rapidly, reliably, and with minimal manual intervention.

Deliverables - specific outputs, products, or results provided as part of the contract or a contractual agreement.

End of life (EOL) - refers to the date after which a product will no longer be sold or renewed (though still might receive some form of support, such as security patches).

End of support (EOS) - refers to the date of complete cessation of all support services for the product, including new patches, updates or fixes.

Infrastructure-as-a-code (IAC) - is an IT practice that involves managing and provisioning computing infrastructure through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

IT ecosystem - is a network of interconnected technologies, software, hardware, and services that work together to support an organization's digital operations. It includes applications, infrastructure, cloud services, and networks, all integrated to ensure seamless communication, scalability, security, and efficiency.

IT solution continuity - involves a collection of strategies and plans focused on maintaining an organization's essential operations during and after disruptions. The goal is to minimize downtime, limit financial losses, and safeguard the organization's reputation when faced with interruptions.

IT solutions - services, products and processes that use information technology to solve business problems, improve operational efficiency, and enhance overall performance. IT solutions are typically composed by various parts. For Example, the codebase, CI/CD routines, documentation etc.

Open-ended - refers to components or features that are flexible, adaptable, and capable of evolving over time to meet a wide range of needs and requirements.

Pre-processing - refers to the series of operations performed on raw data to prepare it for further analysis and processing.

REST API service - is a type of web service that allows systems to communicate over HTTP by accessing and manipulating resources using standard methods such as GET, POST, PUT, and DELETE. RESTful APIs are stateless, meaning each request from a client to the server must contain all the necessary information, and they typically return data in formats like JSON or XML. This approach is widely used for building scalable, lightweight web services and enabling seamless integration between different systems.

Software development tools - applications, frameworks, and utilities that software developers use to create, debug, maintain, or support software.

Software product - a collection of computer programs, procedures, and documentation that performs a specific task or function or provides a comprehensive solution to a particular problem.

Source code - a set of instructions and statements written by a programmer using human-readable programming language. It is the original code written and saved in files before being compiled or interpreted into executable programs. Source code serves as the blueprint for creating software applications.

Workflow - systematic sequence of processes and activities.

4. Principles

The principles are grouped into 8 overarching IT architectural themes:

1. **Architecture** - foundational and design principles for sound infrastructure and IT solution architecture.

2. **Reproducibility** - the overarching goal is to ensure that any deliverable in the form of an IT solution may be reproduced given sufficient time.

3. **Reusability** - the services and products provided through the EEA CLMS are to be reused by the end users as the foundation of their further work.

4. **Transparency** - the EEA CLMS program is funded by the EU and supports its community with data and products. These products are to be part of the foundation for further work in the field and hence transparency is key to ensuring that extended work can be carried out.

5. **Maintainability** - the EEA aims in the CLMS program to be able to provide updated products when new data becomes available e.g. on yearly basis. To reduce the time to market the principles of maintainability is to be followed.

6. **Observability** - IT solutions of the EEA CLMS must collect relevant metrics for monitoring and assessment, to avoid disruptions and have predictable operation of the solutions.

7. **IT security** - IT solutions of the EEA CLMS are utilized by multiple stakeholders and thus IT security is paramount. This is especially critical for the open-ended aspects such as the use of APIs and outward facing web solutions.

8. **Resilience** - IT solutions of the EEA CLMS are designed to withstand and recover from disruptions by remaining operational during unforeseen events.

Together, these principles guide design, development, and evolution of the IT solutions in the EEA CLMS program. The principles should be periodically reviewed and updated to ensure alignment with the latest technological advancements and emerging best practices. This ongoing evaluation will help maintaining the relevance, effectiveness, and security of the IT solutions.

4.1 Architecture

Foundational and design principles for maintaining sound infrastructure and IT solution architecture. These sub-principles addresses best-practices and industry standard design patterns.

Architecture 1:	**Client specific IT solutions should have a modular structure**
What:	Modular structure of client specific IT solutions is a requirement. This may be achieved using e.g. microservice architecture
Why:	A modular structure is sought to ensure further development, and updates are possible. The possibility of substituting or adding modules in an IT solution will increase the lifespan of a solution and increase scalability
Consequence:	Modular architecture of IT solutions is a requisite
Example:	*If the client specific IT solution has, for Example, grown its user base since the launch of the solution, then scaling up shall be possible at any time – scaling containers, vertically (more CPUs, RAM) and horizontally (more VMs)*

Architecture 2:	**IT solutions are to be Dockerized or similar**
What:	The use of container technology is encouraged
Why:	Containerization is crucial for building scalable IT solutions and container technology eases the work of moving IT solutions around the IT infrastructure making deployment easier to automate
Consequence:	IT solutions are to be deployed using Docker containers or similar
Example:	*Software components of the client specific IT solution shall be provided as docker containers so that deployment is flexible with respect to hardware*

Architecture 3:	**Client specific IT solutions must be able to interface with other IT solutions**
What:	The IT deliverable must be able to be used in conjunction with other deliverables to

Architecture 3:	Client specific IT solutions must be able to interface with other IT solutions
	form a composite solution
Why:	To make the most of the funds available the developed solutions should form part of an IT ecosystem making up a whole
Consequence:	IT deliverables must be equipped with documented APIs for interfacing with other IT applications
Example:	*A client specific product, which can be used for extracting and manipulating data, should be accessible programmatically through e.g. well documented REST services*

Architecture 4:	IT solutions should be cloud agnostic
What:	If the IT solution is built for cloud environments, measures must be taken to make the solution cloud agnostic.
Why:	Vendor lock-in must be avoided to remove vendor specific dependencies, making the IT solution easier to migrate to a different cloud vendor
Consequence:	IT solutions must minimize the usage of vendor specific functionality and non-standardized infrastructure
Example:	*An IT solution that makes use of serverless functions should be built in a way that allows for using another vendors serverless functionality with little or no changes in case of migrating from one platform to another*

4.2 Reproducibility

The overarching principle of reproducibility is further unfolded below in the following sub-principles:

Reproducibility 1:	Description of workflows must be provided
What:	Deliverables which are a result of pre-processing of data must be provided with a description of the workflow for the pre-processing
Why:	To ensure that the deliverable can be re-produced, details must be provided on how this can be achieved
Consequence:	Descriptions of pre-processing workflow steps are to be provided with deliverables. Ideally the workflows delivered as scripts or similar. At a minimum documentation of how the workflows are to be set up is to be provided
Example:	*A delivery that includes a web application, shall include description of the build process, such as the compilation of source code, packaging of the application, and deployment steps. This for instance could include details on the specific versions of tools used (e.g. Node.js, Docker etc.)*

Reproducibility 2:	Data sources to be supplied with deliverables
What:	IT solutions which utilize data sources must supply the data sources
Why:	To ensure that the deliverable can be re-produced details are required on data sources used along with any enrichment which have been applied to the data source
Consequence:	Data source location must be provided if data are publicly available. If data are not accessible to the CLMS, the data are to be provided as part of the deliverable
Example:	*If the software relies on a proprietary weather data API that is not publicly accessible, the data, or at least a sample dataset, should be provided with the delivery. If the API is publicly available, detailed instructions on how to access it (e.g.,*

Reproducibility 2:	Data sources to be supplied with deliverables
	API keys, endpoint URLs) must be included

Reproducibility 3:	List of software used in development of IT solution to be provided
What:	The software products which have been used in the development of the software are to be listed as part of the deliverable
Why:	To ensure that the IT solution can be further developed details are required of the software components/products that were used in the development
Consequence:	List of software development tools used in the production to be provided. Further for client specific developments the source code must also be provided
Example:	*A system consisting of several building blocks, such as User Interface, backend, importer, and exporter modules, shall be provided with a list of software development tools, used for production of these building blocks and modules*

Reproducibility 4:	Automation tool/scripts used in the production of the IT solution must be provided
What:	IT solutions which include automation scripts/workflows in the development must supply these scripts as part of the deliverable
Why:	Automation scripts used in development are viewed as part of the deliverable and are required for reproduction of the solution
Consequence:	Automation scripts whether as stand-alone scripts or as a configuration of standard/commercial software must be provided as part of the deliverable
Example:	*If the IT deliverable includes an automatic backup that generates full backups in certain increments, then the automation scripts behind the backup generation must be provided as part of the deliverable, so that they could be recreated*

Reproducibility 5:	If a solution includes outcomes of pre-executed algorithms the prerequisites for running the algorithms must be provided
What:	To ensure reproducibility, the algorithms must be provided either as pseudo code or as source code
Why:	The foundation of the IT solution must be re-producible to ensure future enhancements are possible say if new insights/data become available also after the end of the contractual agreement
Consequence:	Supplier must as part of the deliverable also detail any algorithms which form the basis of the solution
Example:	*A spatial product, providing a detailed pan-European wall to wall 10-meter spatial resolution raster product, that is based on a supervised classification of satellite image time-series. The supplier must provide a detailed description of the algorithm that was used for classifying satellite-imagery time-series*

4.3 Reusability

The principle of reusability is detailed in the following sub-principles:

Reusability 1:	IT solutions should be open-ended equipped with APIs
What:	IT solutions should be open-ended equipped with APIs through which functionality or data key for the end user may be accessed
Why:	To ensure that further work benefit from existing solutions it is paramount that systems delivered are open ended. Future work can hereby utilize and benefit from previous work. Delivered IT solutions should form part of the overall IT ecosystem of the EEA CLMS program so that the "whole is greater than the sum"
Consequence:	IT solutions should be provided with APIs which access key functionality of the IT solutions
Example:	*A webservice provided, which publishes geospatial data, has an API Rest service, which grants users direct access to the data*

Reusability 2:	Scripts used in production must be delivered as part of IT solutions
What:	Scripts should be delivered with code so that they may be used as templates for the end user for further development
Why:	Data, conditions, or requirements may change for an IT solution. To ensure that such changes can be accommodated the underlying script must be possible to modify to reflect and support such changes
Consequence:	Scripts used in the productions form part of the final deliverable
Example:	*IT delivery, consisting of several building blocks, shall be provided with scripts, included with the final delivery of the code, so that the end users of the system could modify, expand, or adopt the building blocks/modules to suit specific needs or add new features*

4.4 Transparency

The CLMS is funded by the EU and supports its community with data and services. As such, these products and services are to be part of the foundation for further work in the field and accessible to the community. To support this, the principle of transparency is detailed in the following subprinciples:

Transparency 1:	Source code of client specific software to be supplied with IT solution
What:	Source code of client specific IT solution is supplied as part of the deliverable and made publicly available under the EUPL-1.2[1] license
Why:	To ensure transparency, it is essential to have clear insights into the client-specific software. This enables efficient future developments and modifications
Consequence:	Source code of client specific software must be delivered with IT solution. The source code shall include Docker recipes and scripts for building the source code and be published under the EUPL-1.2. license
Example:	*Source code of all the components of the specific IT solution must be delivered. Any updates or developments of the source code shall be reflected*

[1] European Union Public Licence - European Commission (europa.eu)

Transparency 1:	Source code of client specific software to be supplied with IT solution
	in the EEA GitHub repository, which is the main repository of the system. Moreover, the specific client IT solutions shall be published under the EUPL-1.2 license, so the openness and transparency are ensured

Transparency 2:	Inline documentation of the source code
What:	Source code of client specific IT solution must be documented in-line
Why:	To effectuate the handover from one developer to the next inline documentation are to be included to guide the developer on the job
Consequence:	Source code must have inline documentation. Inline code should be formatted so that it may be easily extracted to generate online documentation
Example:	*Source code of all the components of the IT solution must have inline documentation. The documentation shall be structured, following common conventions, and kept at a minimal, but comprehensive level*

Transparency 3:	Commercial software used in the production must be attainable by the EEA or a third-party provider
What:	Commercial software which are prerequisites must be attainable on comparable terms. Such software is justified only if no open alternative exists
Why:	To ensure that further work may be carried out any prerequisites in the form of software must be attainable by the EEA or a supplier
Consequence:	Generally attainable commercial software used in production must be listed when delivering an IT solution. Name of software, version, EOL and EOS to be supplied
Example:	*An IT solution is deploying various components and the set-up of the virtual machines that houses the components is done by means of an infrastructure as-a-code-tool. All the capabilities of the infrastructure as-a-code-tool, that require purchasing must be listed when delivering the IT solution*

4.5 Maintainability

The EEA aims in the CLMS program to be able to provide updated products when new data becomes available. To reduce the time to market the principle of maintainability is to be followed.

Maintainability 1:	IT solutions are to be delivered on a principle of CI/CD
What:	The launch of new releases of IT deliverables are to be configured and managed so that new functionality is available as soon as possible. The principle of CI/CD are to be adhered to
Why:	Time to market is to be reduced through an approach of maintainability of deployments as soon as possible
Consequence:	IT deliverables are to be supplied with a dev-ops set-up which supports CI/CD
Example:	*A delivered IT solution is organized with a test server environment potentially a pre-production environment, used for quality assurance and continuous development, so that deployment to production can be initiated smoothly*

Maintainability 2:	Tests are to be organised so that they may be automated
What:	Tests are to be structured so that they may be easily automated
Why:	To ensure that the build and CI/CD process does not introduce bugs or deployment failures, tests are to be automated so that they can continuously be run to ensure the quality of the solution and its possible enhancements
Consequence:	Tests are to be delivered so that they can be automated
Example:	*The delivered solution has in the test phase run through a number of tests e.g. unit tests and result verification tests. These will be the basis for automated regression tests*

Maintainability 3 (Reusability 3):	Documentation of IT solutions are to be provided
What:	Documentation of the developed IT solutions must be provided. The requested documentation shall also be provided in quarto markdown format on the dedicated EEA GitHub repository
Why:	For the further use and improvements of the IT solution, technical documentation is paramount.
Consequence:	Documentation including but not limited to System Description Document (SDD), System Deployment Document and Examples must be provided with IT solution deliverables. The requested documents shall also be provided in the quarto markdown format.
Example:	*An IT delivery, consisting of several building blocks, shall be provided with SDD, user guidelines, and detailed documentation of system deployment, including, but not limited to system and storage architecture, infrastructure setup, provisioning, monitoring, disaster recovery, accessibility, scalability options and performance. If requested, this documentation shall be provided in quarto markdown format on the dedicated EEA GitHub repository*

Maintainability 4:	Commercial software used in the development must be attainable by the EEA or a third-party provider
What:	Commercial software which are prerequisites must be attainable on comparable terms. Such software is justified only if no open alternative exists
Why:	To ensure that further work may be carried out any prerequisites in the form of software must be attainable by the EEA or a supplier
Consequence:	Generally attainable commercial software used in development or production must be listed when delivering an IT solution. Name of software, version, EOL and EOS to be supplied
Example:	*An IT solution using commercial components or tools, like PDF generator, code analysis tools, data transformation software must be listed*

Maintainability 5:	Deployment and integration scripts of client specific software to be supplied with IT solution
What:	Deployment and integration scripts of client specific IT solution is supplied as part of the deliverable

Maintainability 5:	Deployment and integration scripts of client specific software to be supplied with IT solution
Why:	To ensure transparency and efficient maintainability, it is essential to have clear insights into the build and deploy processes of client-specific software. This enables efficient future developments and modifications
Consequence:	Scripts or playbooks and documentation for CI/CD (Continuous Integration/Continuous Development), Docker recipes and build scripts must be delivered
Example:	*Source code of all the components of the specific IT solution must be delivered. Any updates or developments of the source code shall be reflected in the EEA GitHub repository, which is the main repository of the system. Moreover, the specific client IT solutions shall be published under the EUPL-1.2 license, so the openness and transparency are ensured*

4.6 Observability

IT solutions of the EEA CLMS must collect relevant metrics for monitoring and assessment, to detect any issues and have predictable operation of the solutions.

Observability 1:	IT solutions are to be regularly assessed
What:	IT solutions are to be automatically monitored with a notification service, and their performance routinely evaluated to ensure optimal functioning
Why:	Regular assessments ensure that IT solutions can be maintained so as to meet emerging needs, threats and technological advancements
Consequence:	IT solution's scalability, security, and overall performance are continuously monitored and evaluated to address performance and security issues
Example:	*The delivered IT solution and its associated dependencies are regularly assessed and evaluated. The evaluation process should also account for advancements in technology and track developments to ensure the solution remains relevant and effective*

Observability 2:	Continuous monitoring of metrics
What:	IT solutions logs metrics on it's components and containers for tracking system performance and application health
Why:	Continuous monitoring gives a data-driven insight of a solutions components performance and health and provide the metrics for automatically scaled solutions and self-recovering solutions
Consequence:	Components and containers in the solution logs relevant metrics to be collected and monitored. As minimum liveliness and readiness should be logged
Example:	*A software solution with an orhcestrating component and a worker component use liveliness and readiness to monitor if the solution is healthy and automatically scale the number of worker instances according to the readiness metrics*

4.7 IT security

The IT solutions of the CLMS program shall ensure system integrity against various security threats, protection of the data, and maintenance of privacy. The following sub-principals are to be followed:

IT security 1:	**Incorporate security considerations from the beginning of the system development**
What:	Ensure security is integrated into all stages of the system development lifecycle, from planning to deployment
Why:	Early integration of security measures reduces vulnerabilities, lowers costs associated with late-stage fixes, and ensures robust protection against threats
Consequence:	Threat modelling and security assessments need to be conducted from the start, as well as allocation of resources for ongoing security reviews and testing
Example:	*Standard aspects such as two factor authentication, protection against SQL injection, encryption of sensitive data, no root users in containers, etc.*

IT security 2:	**Compliance with relevant laws, regulations and industry standards**
What:	IT-solutions must adhere to legal requirements, industry standards, and regulations e.g. EUDPR, ISO
Why:	Compliance ensures legal and regulatory adherence, builds trust, protects sensitive data, and mitigates risk of legal penalties and breaches
Consequence:	IT deliverables need to incorporate robust security measures, include documentation of compliance efforts, and ensure features and processes aligned with legal and industry measures
Example:	*Data handling agreements must be in place, consideration of server location in EU, etc.*

IT security 3:	**Ensuring that users and systems have appropriate permissions based on their roles and responsibilities**
What:	Implement role-based access control (RBAC) to manage user and system permissions according to their roles
Why:	It prevents unauthorized access, minimizes the risk of data breaches, and ensures that users only have access to the information necessary for their roles
Consequence:	The provider will need to define clear roles and responsibilities, implement RBAC policies, regularly review and update access controls
Example:	*A delivered IT solution has role-based accesses, which ensures that only Admin-Users are allowed to manage (add, edit, activate, inactivate) users and organisations. Also, only administrator can view and edit any ingestion and extraction within the system to support users if they need any help*

IT security 4:	**Logging warnings and errors**
What:	The IT solution must log all errors, warnings and events with audit relevance from every component to a file based storage
Why:	In order to inspect system events and detect potential security incidents is crucial for maintaining the system integrity and resilience. Log information must not be revealed to the user, but must be stored internally.
Consequence:	All components of an IT solution must log audit, error and warning information coming from executing the code of the solution
Example:	*A user logs in to an application, trying to download a large dataset for processing in*

IT security 4:	Logging warnings and errors
	the application, the system encounters some fatal errors with the download. Login, user activity and technical error information and severity is logged to a persistent file storage.

4.8 Resilience

Resilience 1:	IT solution should have a disaster recovery plan
What:	IT solution should have a well-defined process of restoring IT systems, data, and operations following a disruption
Why:	To ensure that the IT solution and data are recoverable after an unforeseen event
Consequence:	IT deliverables will be provided with well-prepared disaster recovery plan that will ensure a rapid restoration of services and data integrity, and minimize damage
Example:	*A delivered IT solution has a disaster recovery plan that includes backup protocols, data replication, and recovery timelines*

Resilience 2:	Ensuring IT solution continuity
What:	IT solution is designed and implemented in a way that ensures continuous operation during a disruption
Why:	To maintain critical operations with a minimal downtime, even when confronted with unforeseen events
Consequence:	IT deliverables are designed for high availability, incorporating redundancy so that in case of a disruption/failure, restore service can immediately take over, minimizing downtime and ensuring continuous operation
Example:	*In the event of a system failure or disruption of the delivered IT solution, restore service automatically take over to maintain service continuity. For instance, if a primary system goes down, a secondary system activates, ensuring that users experience no downtime.*