VRE4EIC

A Europe-wide Interoperable Virtual Research Environment to Empower Multidisciplinary Research Communities and Accelerate Innovation and Collaboration

Deliverable D3.1

Architecture Design

Document version: 1.0

VRE4EIC DELIVERABLE

Name, title and organisation of the scientific representative of the project's coordinator:
Mr Philippe Rohou t: +33 4 97 15 53 06 f: +33 4 92 38 78 22 e: philippe.rohou@ercim.eu
GEIE ERCIM, 2004, route des Lucioles, Sophia Antipolis, F-06410 Biot, France
Project website address: http://www.vre4eic.eu/

Project	
Grant Agreement number	676247
Project acronym:	VRE4EIC
Project title:	A Europe-wide Interoperable Virtual Research Environment to Empower Multidisciplinary Research Communities and Accelerate Innovation and Collaboration
Funding Scheme:	Research & Innovation Action (RIA)
Date of latest version of DoW against which the assessment will be made:	14.01.2015
Document	
Period covered:	M1-M12
Deliverable number:	D3.1
Deliverable title	Architecture Design
Contractual Date of Delivery:	September 30, 2016
Actual Date of Delivery:	October 1, 2016
Editor (s):	Carlo Meghini (CNR ISTI)
Author (s):	Keith Jeffery (ERCIM) Cesare Concordia, Eda Marchetti (CNR ISTI) Theodore Patkos, Nikos Minadakis, Yannis Marketakis, Ioannis Chrysakis (FORTH ICS) Jacco van Ossenbruggen, Jan Wielemaker (CWI)
Reviewer (s):	Daniele Bailo (INGV), Laura Hollink (CWI)
Participant(s):	
Work package no.:	3
Work package title:	Architecture, VRE development, integration and scalability
Work package leader:	Carlo Meghini (CNR ISTI)
Distribution:	Confidential, VRE4EIC and EC only
Version/Revision:	1.0
Draft/Final:	Final
Total number of pages (including cover):	89

What is VRE4EIC?

VRE4EIC develops a reference architecture and software components for VREs (Virtual Research Environments). This e-VRE bridges across existing e-RIs (e-Research Infrastructures) such as EPOS and ENVRIPLUS, both represented in the project, themselves supported by e-Is (e-Infrastructures) such as GEANT, EUDAT, PRACE, EGI, OpenAIRE. The e-VRE provides a comfortable homogeneous interface for users by virtualising access to the heterogeneous datasets, software services, resources of the eRIs and also provides collaboration/communication facilities for users to improve research communication. Finally it provides access to research management /administrative facilities so that the end-user has a complete research environment.

Disclaimer

This document contains a description of the VRE4EIC project work and findings.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any responsibility for actions that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the VRE4EIC consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 28 Member States of the Union. It is based on the European Communities and the Member States cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors (http://europa.eu/).

VRE4EIC has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676247.

Table of Contents

1 Introduction 9
2 Methodology and approach 11
2.1 Elicitation and collection of requirements 11
2.2 Functional architecture design 12
2.3 Architecture design 12
2.4 REFERENCES 13
3 Analysis of requirements 14
4 The VRE4EIC Reference Architecture 15
4.1 Rationale 15
4.1.1 Virtual Research Environment 15
4.1.2 A Vision for the e-VRE Architecture 16
4.2 Component Overview 20
4.3 Components in detail 22
4.3.1 User Manager component 22
4.3.2 AuthenticatorApp: *Authentication* interface description 25
4.3.3 Resource Manager component 26
4.3.4 Workflow Manager Component 28
4.3.5 Message Oriented Manager (MOM) component 30
4.3.6 Resource Adapter 31
4.3.7 APP Manager Component 31
4.3.8 Metadata Manager (MM) Component 32
4.3.9 The Query Manager (QM) Component 36
4.3.10 The Model Mapper Component 39
4.3.11 The LD Manager Component 41
4.3.12 eVRE WS component 43
4.3.13 AAAI Component 44
4.4 References 45
5 Joining an eVRE instance 46
6 Assessment of the architecture 48
6.1 Introduction 48
6.1.1 VREs 48
6.1.2 e-RIs 49
6.1.3 e-Is 49
6.2 Use Case Sequence Diagrams 49
6.2.1 Search (simple/advanced) 50
6.2.2 Cross search 52
6.2.3 Results browsing 54
6.2.4 Data publishing 56
7 Outlook 58

8 Conclusions 59
9 Annexes 60
9.1 Architectural components 60
9.2 Generalised functions 62
9.3 Requirements and components 76

Table of Figures

Figure 1 Entities and relationships involved in the analysis of requirements 14
Figure 2 Resources, e-RIs Services and VRE Services 16
Figure 3 Alternative approaches to the cooperation on e-RIs and a VRE 17
Figure 4 Architectural tiers in a VRE 18
Figure 5 Conceptual components and logical tiers 19

Executive Summary

This deliverable presents the initial Reference Architecture of the VRE4EIC Project.

The Reference Architecture is one of the results of the first year of work of the project. In order to derive it, the project conducted more than sixty interviews, and carefully characterized five existing e-Research Infrastructures. This way, it obtained a significant picture of the needs of the scientists, and of how these needs are met by the existing e-Research Infrastructures.

This picture was an input to the architects of the Project, who analysed it in detail, and distilled an exhaustive list of functions whose implementation is needed to satisfy the requirements in ways that advance the current e-Research Infrastructures.

Competence in software system design and development, as well as consideration of existing standards and best practices, allowed the VRE4EIC architects to group functions into components, and to further structure components in sub-components, thereby laying the basic building blocks of the Reference Architecture. Each component or sub-component was assigned a set of interfaces, each consisting of methods performing the tasks that provide the lowest level of granularity of the Reference Architecture.

An intense activity of verification then ensued, aiming at verifying that the methods on the component interfaces could be appropriately composed into complex workflows able to realize the required functionality.

The deliverable presents the result of this work, and documents the activities and the decisions that led to it.

Based on this solid piece of work, the project is entering its second year of activity. While keeping the Reference Architecture aligned with additionally collected requirements and e-RI characterizations, a gap analysis will be conducted to assess two existing Virtual Research Environments with respect to the Reference Architecture. The gap analysis will allow the project to select the components of the Reference Architecture that will be implemented in order to enhance the selected Virtual Research Environments.

As a result of these activities, at the end of the project the final Reference Architecture will be released.

1 Introduction

This deliverable provides the initial architectural design of e-VRE; the virtual research environment reference architecture (later to be deployed as a prototype). The architectural design has been produced utilising:

1. Requirements elicited by interviews of representatives of various RIs (Research Infrastructures) – in particular of their e-RIs electronic representation as systems for access and utilisation (D2.1);
2. Characteristics of the e-RIs in terms of assets available (users, data, software components, services (including workflows), resources (computers, detectors, instruments) (D2.1 live document);
3. Analysis of existing VREs, SGs (Science Gateways) and VLs (Virtual Laboratories) internationally to cross-verify the requirements and characteristics.

The design method refines from business architecture (based on business requirements) through functional architecture, applicative architecture and technical architecture. The refinement has included input from:

1. WP4: on the metadata mappings to allow information from the catalogs of e-RIs to be included in that of e-VRE to allow optimal interfacing between e-VRE and each e-RI in the dimensions of users, data, software components, services (including workflows) and resources (including computers, detectors, equipment);
2. WP5: on the integration of policy aspects or NFRs (non-functional requirements) covering trust, security, privacy, rights (including licensing), SLA (service level agreements) and QoS (quality of service) implying performance, scalability and reliability to ensure the NFRs are pervasive through the software stack not only in the e-VRE but also the accessed and utilised e-RIs (including their e-I (e-infrastructure) platforms);

The result is the specification of a set of components of the e-VRE with their purpose, function and interfaces defined and their inter-relationships (internal and external) specified. Particular attention has been given to ensuring generality - since the object is a reference architecture that can be specialised for any given domain while retaining the core features for interoperability – while also ensuring the possibility of evolution to meet changing requirements and changing opportunities arising from new technology. Furthermore, the architecture design has been positioned in the ecosystem of e-Is (such as GEANT, AARC2, EUDAT, PRACE, EOSC, OpenAIRE, and others) and the e-RIs especially those in the ESFRI (European Strategic Forum for Research Infrastructures) roadmap document.

This deliverable leads to the gap analysis (D3.2 M15), the software building blocks (D3.3 M24), and the specification of enhanced VREs (D3.4 M30). This deliverable forms the basis for development of the prototype and – based on that experience – the final architecture deliverable (D3.5, M36) will be produced. In order to strengthen this first effort, we have also carried out an assessment of the Reference Architecture, based on the requirements collected so far. The assessment aims at validating the Reference Architecture with respect to the analysis that led to it: it consists in a series of sequence diagrams of the most significant high-level use cases. These sequence diagrams show that the set of components that we have derived at this stage, along with their interfaces, is able to respond to the use cases.

The deliverable is structured as follows:

- Section 2 presents an overview of the general methodology that has been applied in order to produce the Reference Architecture, and of the specific approach that has been followed in applying the methodology in the context of the VRE4EIC project;
- Section 3 presents the analysis of requirements that has been performed to derive the components of the Reference Architecture and their interfaces;
- Section 4 gives the Reference Architecture as a set of UML component diagrams, highlighting the interfaces provided and used by each component, and the methods that constitute these interfaces. Each method is documented by its signature and a description of its behavior;
- Section 0 discusses the steps that an e-RI must take in order to join an existing e-VRE, running an instance of the Reference Architecture;
- Section 6 presents an assessment of the Reference Architecture by relating the Reference Architecture with ongoing work in the area of VREs and by presenting sequence diagrams realizing the most important use cases taken from deliverable D2.2; overall, these show the adequacy of the Reference Architecture from a functional point of view.
- Section 7 outlines the further development of the Reference Architecture within the VRE4EIC project.
- Section 8 concludes.
- The Annex reports tables that document the steps of the architecture derivation process.

For compactness, references are resolved at the end of the Section where they occur.

2 Methodology and approach

In literature, software architecture development targets the definition of a structured solution able to satisfy the functional and non-functional requirements of an application domain. Thus the development of a software architecture involves different points of views: the user, the system (the IT infrastructure), and the business goals. For each of these areas, the key scenarios/requirements should be identified as well as quality attributes. Then requirements should be refined into architectural functions and mapped into specific architectural components or modules.

This process raises a series of criticalities and issues and should be carefully developed so as to avoid architectural failures or incompleteness. For this reason, in the last 10 years different approaches for architecture specification have been proposed, largely inspired from the Software Development Life Cycle, SDLC [ISO/IEC 12207]. This is a well-defined, structured sequence of stages targeting the specification and the successive development of the intended software product. It involves a series of decisions based on a wide range of factors, and each of these decisions can have considerable impact on the quality, performance, maintainability, and overall success of the application.

A typical SDLC includes different activities such as: understanding of business needs and constraints; elicitation and collection of requirements; functional architecture design; architecture design; implementation; testing; deployment; maintenance.

The order in which these activities are to be executed is usually defined into a specific software development process such as Waterfall model, incremental model, RUP, V-model, iterative model, RAD model, Agile model, Spiral model, Prototype model, to mention just a few [SE].

In the development of the Reference Architecture reported by the present deliverable, an incremental software development process largely inspired by the RUP process [RUP, UP] has been followed. In this Section, the main characterisation of the process to the specific exigencies of the project constraints and activities are schematised. In particular, the Section provides details about activities concerning the software architecture specification and design that are: elicitation and collection of requirements; functional architecture design; architecture design.

The process characterisation has been performed in collaboration with the different partners, considering the output of the other deliverables (see previous Section) and analysing the current available proposals of VREs. As a consequence, the Reference Architecture specification provided in this deliverable will document two views:

1. Component diagram: it describes the components necessary to implement the eVRE functionalities. It visualizes the physical components in a system as well as the interfaces among them.
2. Interaction diagrams: they describe the type of interactions among the different components of the architecture and represent the part of dynamic behavior. We consider in particular sequence diagrams which emphasize on time sequence the message exchange

In the remaining of this Section, the three main activities of the SDLC specifically characterized for the eVRE architecture specification are presented.

2.1 Elicitation and collection of requirements

Elicitation and collection of requirements is a fundamental stage in the eVRE architecture specification. From a technical point of view, it involves different stages such as [ISO/IEC/IEEE 29148]:

1. High-level use case definition: A representative of the stakeholder community presents the business/mission drivers for the eVRE considering also quality attributes, security aspects.

2. Use case definition: Stakeholders express scenarios representing their concerns about the system prioritizing when possible the main ones.
3. Specification eVRE requirements: the main use cases are analysed and refined in more detail. The focus is on specifying what a system should do (the functional requirements) and on how the system should function (the non-functional, or quality, requirements) [ISO/IEC 25010, ISO/IEC 25030].

In the context of the VRE4EIC project, this stage has been carried out by task 2.1. The Reference Architecture development team has acquired the results of the stage in two different forms, each at a different time. In particular,

- eVRE Requirements have been made available early in the development, and therefore they have been used to derive the functional design, as illustrated next;
- High-level use cases and have been made available later, and therefore they have been used to validate the Reference Architecture, as illustrated in Section 6.

2.2 Functional architecture design

During this activity, the identified requirements are analysed and one or more architectural functions are derived. For assessment purposes, requirements mapping on the derived functions is also performed.

In the context of the VRE4EIC project, this stage has been carried out by Task 3.1, as the first stage of its development, and is documented in the next Section of this deliverable.

During its execution, several design guidelines have been followed [RUP, https://msdn.microsoft.com/en-us/library/ee658124.aspx]. A synthesis of the most important ones is provided below.

- *Separation of functions*: Isolate from the requirements different functions with as little overlap in functionality as possible. The important factor is minimization of interaction points to achieve high cohesion and low coupling.
- *Aggregation of functions:* Identify possible generalization or composition relations between the isolated functions to improve the organization and readability of the functional architecture design.
- *Learn from similar projects:* analyze similar projects and documentation so to derive an high conceptual-level architecture description focusing mainly on high view of modules/components and communications and interactions between them.
- *Reduce Responsibility:* Assign to each component or module the responsibility for only a specific functionality or aggregation of cohesive functionality.
- *Minimal Knowledge:* Each component or module should be unaware of the internal details of other components.

2.3 Architecture design

The functional architectural design should be refined and further decomposed so that identified functional and nonfunctional requirements are completely satisfied. In particular, the allocation of each of the identified functions to a component has to be completed. During this activity the interfaces of the components, modules and sub-modules are also defined and documented.

In the context of the VRE4EIC project, this stage has been carried out by Task 3.1, as the second stage of its development, and is documented in the Section 4 of this deliverable.

Also for carrying out this step several design guidelines have been followed [RUP, https://msdn.microsoft.com/en-us/library/ee658124.aspx]. Here below a synthesis of the most important ones is provided.

Assess minimal knowledge: verify that each component does not rely on internal details of other components. In particular check that each component method is called from at least another object or component. Verify also that the method has information about how to process the request and, if appropriate, how to route it to appropriate subcomponents or other components.

Avoid overloading of the functionality of a component: Avoid to overloaded components with many functions and applying the single responsibility and separation of concerns principles.

Focus on communication between components: Understand the deployment scenarios and determine if all components will run within the same process, or if communication across physical or process boundaries must be supported—perhaps by implementing message-based interfaces.

Define a clear contract for components: Components and modules should define a contract or interface specification that describes their usage and behavior clearly. The contract should describe how other components can access the internal functionality of the component, module, or function; and the behavior of that functionality in terms of preconditions, postconditions, side effects, exceptions, performance characteristics, and other factors.

The validation of the architecture, also part of this stage, has been carried out as the last stage of D3.1, and is documented in the Tables provided in the Appendix, in the way described in Section 3. In particular traceability relationships able to provide associations between requirements and their realizations are reported. These relationships will help either to assess the percentage/level of requirements implemented into the architectures or to connect the various architecture modules/component to the original requirement [RUP]. More details about the traceability relationship between use cases, requirements and components, are provided in Section 3.

2.4 REFERENCES

[ISO/IEC 12207] International Organization for Standardization, “ISO/IEC/IEEE 12207:2008 - Systems and software engineering -- Software life cycle processes,” ISO/IEC, Mar. 2008.

[ISO/IEC/IEEE 29148] International Organization for Standardization, “ISO/IEC /IEEE 29148:2011 - Systems and software engineering — Life cycle processes — Requirements engineering,” ISO/IEC/IEEE, Nov. 2011.

[SE] Ian Sommerville. Software Engineering. 9th Edition, Addison-Wesley 2011

[RUP] Rational Unified Process Best Practices for Software Development Teams, Rational Software White Paper TP026B, Rev 11/01, July 2003 https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[UP] Jacobson, I., Booch, G., Rumbaugh, J., Rumbaugh, J., & Booch, G. (1999). The unified software development process (Vol. 1). Reading: Addison-Wesley.

[ISO/IEC 25010] International Organization for Standardization, “ISO/IEC 25010 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models,” ISO/IEC, Mar. 2011.

[ISO/IEC 25030] International Organization for Standardization, “ISO/IEC 25030 - Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements,” ISO/IEC, June 2007.

3 Analysis of requirements

As explained in Section 2, the components of the Reference Architecture have been derived based on an analysis of the requirements collected in Task T2.1 of the project. Fig. 3.1 presents a UML class diagram outlining the entities involved in this analysis and their relationships. In particular:

- The analysis started from the Requirements (yellow box in Figure 1). Each requirement has been considered individually, and the functions (green box) required for its implementation have been derived. The association between requirements and functions is documented in Tables 2.1 and 2.2 given in Appendix. In particular, the requirements associated to a function are given in the column labeled as "RequirementID" of these Tables.
- In order to ease the specification of functions, a set of generalised functions has also been derived, which are included or specialised by functions, or which may be used as preconditions by functions. Generalised functions are reported in Table 2 in Appendix, along with their relations to functions.
- The components that are required for the implementation of functions have finally been derived (T3:Components Involved). Components are detailed in Table 1, also given in the Appendix; Table 1 also documents the sub-components derived for each component.

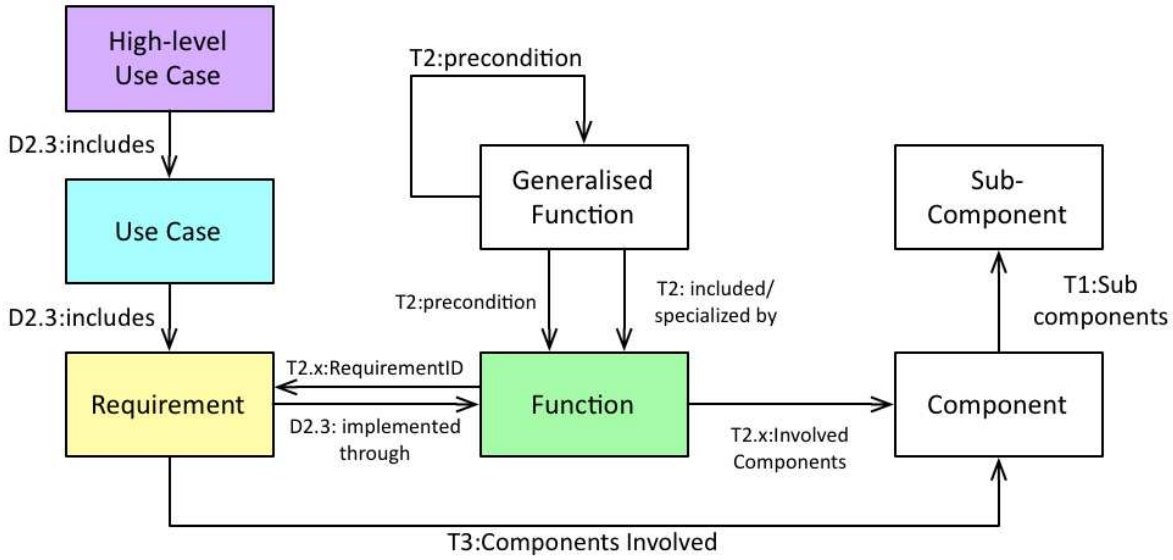


Figure 1 Entities and relationships involved in the analysis of requirements

As Figure 1 also shows, this analysis of requirements into functions and components connects to the use cases (turquoise box) and the high-level use cases (purple box) derived in parallel by Task T2.2 and documented in deliverable D2.3. The connection is realized through requirements, and will be used in Section 6 for the assessment of the Reference Architecture.

Overall, the schema shown in Figure 1 allows us to maintain the relationship between use cases, requirements and components, thereby realizing the traceability of the Reference Architecture.

4 The VRE4EIC Reference Architecture

This Section is the core of the deliverable, reporting the Reference Architecture design. It is divided into two main subsections:

- Section 4.1 gives the rationale of the architecture, moving from an analysis of the notion of Virtual Research Environment into a vision for the e-VRE architecture, consisting of the main components of the Reference Architecture
- Section 4.3 gives the detailed specification of the components derived previously, presenting provided and used interfaces of each component, and the signature of each method in an interface

4.1 Rationale

4.1.1 Virtual Research Environment

In essence, the goal of a VRE system is to decouple Science from ICT complexity, by providing researchers with a facility that takes care of ICT so allowing them to focus on their work. In this sense, a VRE is a fundamental component of an e-RI (e-Research Infrastructure) as it makes the resources of the e-RIs easily accessible and reusable to the community of researchers that owns the e-RI. Here, by e-RI we mean "facilities, resources and related services used by the scientific community to conduct top-level research in their respective fields"[1] while *resource* indicates any ICT entity that is of interest in an e-science community. Typically, a resource is owned by an e-RI that provides an identity for the resource and manages it, making it accessible and re-usable. Examples of resources are: datasets, workflows, algorithms, Web Services, computational or storage facilities, cloud endpoints etc.

In general, a VRE is expected to:

- allow researchers to communicate with each other and to share and use the resources available in the community's e-RI
- allow researchers to advance the state of the art by building new resources as the result of processing existing resources with the available tools. Such processing may be the application of an individual piece of software to a dataset, such as the extraction of certain knowledge from a single file; or, it may result from the execution of a complex workflow obtained by combining available services, including other workflows
- allow research managers to apply economy of scale models to access and manage resources that researchers or single organizations alone could not afford.

Moreover, a VRE can offer all of the above within an individual e-RI or across several e-RIs, the latter option clearly requiring a level of interoperability that would empower researchers and managers in ways that are only imaginable today.

The most advanced e-RIs have developed their own VRE, showing awareness of the crucial role that a VRE can play for their researchers. Others are currently designing their VRE. However, the number of currently existing or designed VREs is very limited; more importantly, these VREs show a great heterogeneity in scope, features, underlying protocols and technologies, partially defeating the interoperability goal that lies at the very heart of a VRE. One of the major goals of the VRE4EIC project is to overcome this issue building an e*nhanced VRE* (e-VRE) system whose main features are:

- Increase the quality of VRE User Experiences (UX) by providing user centered, secure, privacy compliant, sustainable environments on searching data, composing workflows and tracking data publications.

[1] http://ec.europa.eu/research/infrastructures/index_en.cfm?pg=what

- Increase the deployment of the VRE on different clusters of research infrastructures by abstracting and reusing building blocks and workflows from existing VREs, infrastructures and projects.
- Improve the contextual awareness and interoperability of the metadata across all layers of the resources in the VRE.
- Promote the exploitation and standardisation of the VRE4EIC solution to different research domains and communities.
- Provide interoperation across 'silo' e-RIs.

The main step to implement the above features is to create a Reference Architecture that can serve as a guide for the development of interoperable VREs. Indeed, the Reference Architecture is one of the two main outputs of the VRE4EIC project. In the following, we elaborate on the architectural aspects of e-VRE in order to produce a vision that lies at the basis of the Reference Architecture, specified in the rest of this Section.

4.1.2 A Vision for the e-VRE Architecture

In order to enable a VRE to make its resources available to the researchers that use the VRE, each e-RI that *participates* to a VRE must provide *descriptions* of its resources, and such descriptions must be rich enough in information to support the VRE services. This information may include the protocol that must be used to interact with an e-RI service; the schema, size and operations allowed on a e-RI dataset; the permission framework that must be adopted for authentication/authorization of the e-RI users, and so on. This process is naturally divided into two steps, as depicted by Figure 2: the e-RI resources are given at the bottom level since they are the basic assets that both e-RIs and VREs operate on; at the next level up, the descriptions of these resources used by the e-RI services (*e-RI Resource descriptions*) are given, next to the e-RI services using them; at the top level, the descriptions of the e-RI resources used by the VRE services (*VRE Resource descriptions*) are given, next to the VRE services using them. In both cases, the services mentioned are purely exemplificative.

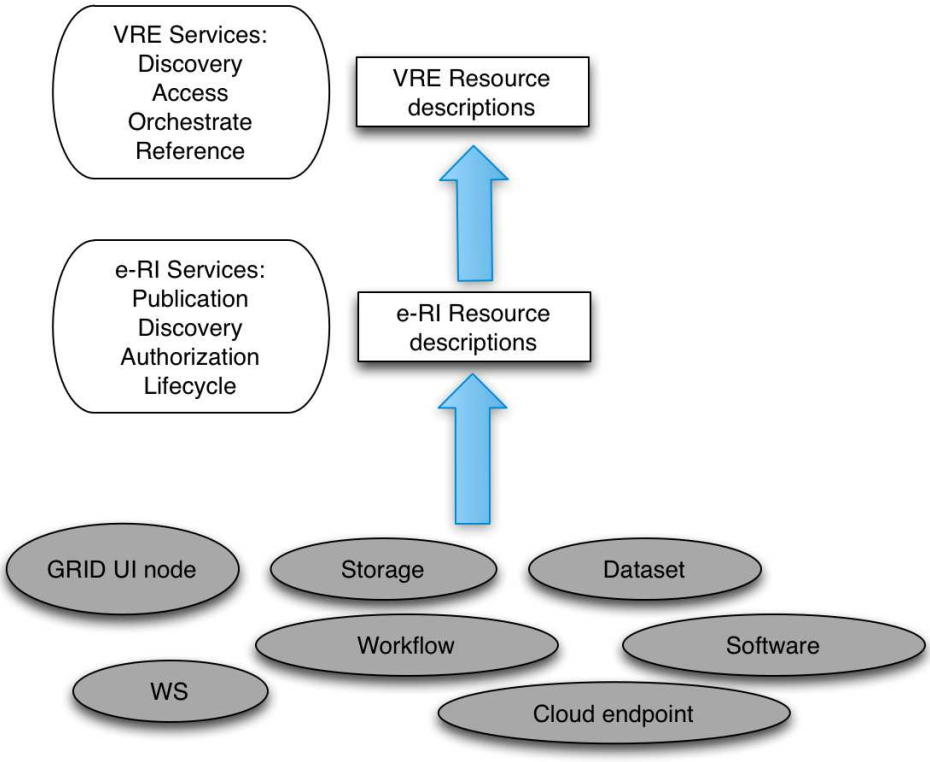


Figure 2 Resources, e-RIs Services and VRE Services

Resource descriptions are typically collected in *Catalogues*. So, the VRE needs to access the catalogues of the participating e-RIs in order to discover the existing resources in the e-RI and obtain enough information on these resources to create its own descriptions of them in its own catalogue.

In order to simplify the creation of the VRE catalogue, one could employ the same data model for both the e-RI and the VRE descriptions. In fact, VREs that live within e-RIs follow this approach. In this case, e-RIs and VRE Resource descriptions only differ for the type of information they contain, while sharing the identity and the basic attributes of resource descriptions. However, this approach is not feasible for VREs with *many* participating e-RIs, due to the fact that in general different e-RIs use different data models to structure their catalogues. In this case, there are two main approaches to create and maintain the VRE catalogue:

- The *centralized* approach (see Figure 3, left), in which there exists a VRE Catalogue used by the VRE services for carrying out their own operations.
- The *distributed* approach (see **Error! Reference source not found.**Figure 3, right), in which there is no VRE Catalogue, but the VRE accesses the e-RIs catalogues when the information is needed.

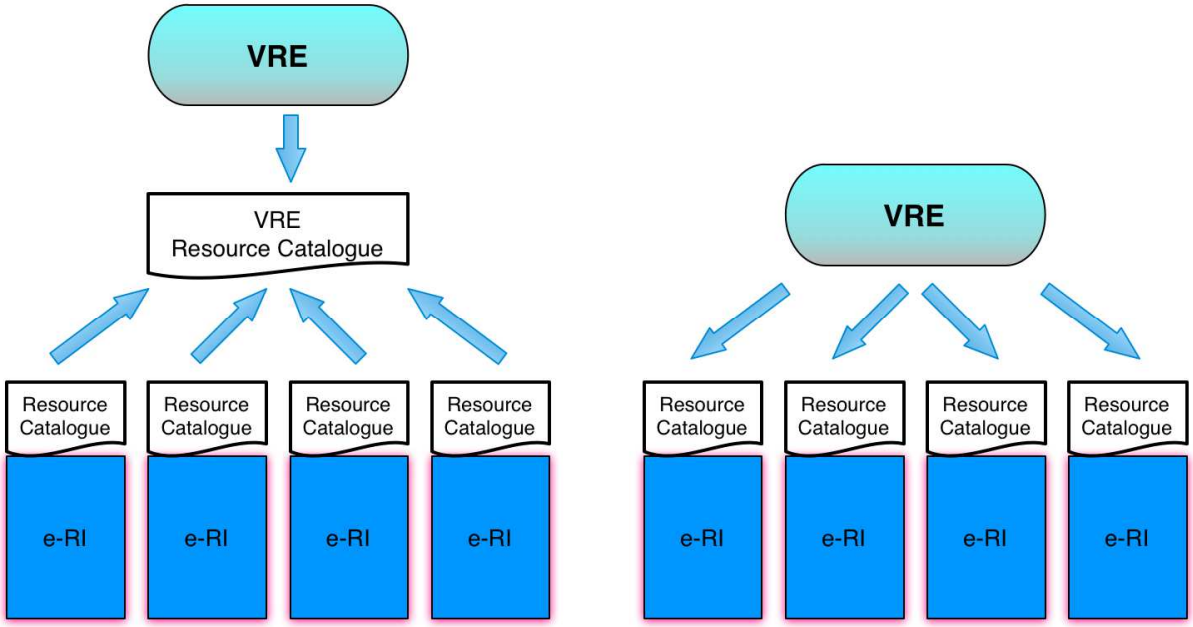


Figure 3 Alternative approaches to the cooperation on e-RIs and a VRE

Each approach has its own pros and cons, as it is well known in distributed system design. In fact, the availability of a VRE Catalogue facilitates all VRE operations that rely exclusively on resource descriptions, such as resource discovery. For operations that require data access, such as data discovery, the centralized approach can only alleviate the problem, by offering information for executing part of the operation. On the other hand, the distributed approach makes it easier to have complete information in real time, since it does not require propagation of updates to the Catalogue.

Our Reference Architecture chooses the centralized approach, because it facilitates one important service, namely the construction of workflows across one or more RIs. The construction of workflows requires numerous access to resource descriptions, followed by optimisation for parallel/distributed operations; the centralized approach makes it possible to implement this access in the most efficient way possible.

At a more general level, the Reference Architecture conforms to the multi-tiers view paradigm used in the design of distributed information systems. Following this paradigm, we can individuate three logical tiers in e-VRE, as shown in Figure 4:

- The *Application* tier, which provides functionalities to manage the system, to operate on it, and to *expand* it, by enabling administrators to plug new tools and services into the e-VRE.
- The *Interoperability* tier, which deals with interoperability aspects by providing functionalities for: i) enabling application components to discover, access and use e-VRE resources independently from their location, data model and interaction protocol; ii) publishing e-VRE functionalities via a Web Service API; and iii) enabling e-VRE applications to interact each others.
- The *Resource Access* tier, which implements functionalities that enable e-VRE components to interact with eRIs resources. It provides synchronous and asynchronous communication facilities.

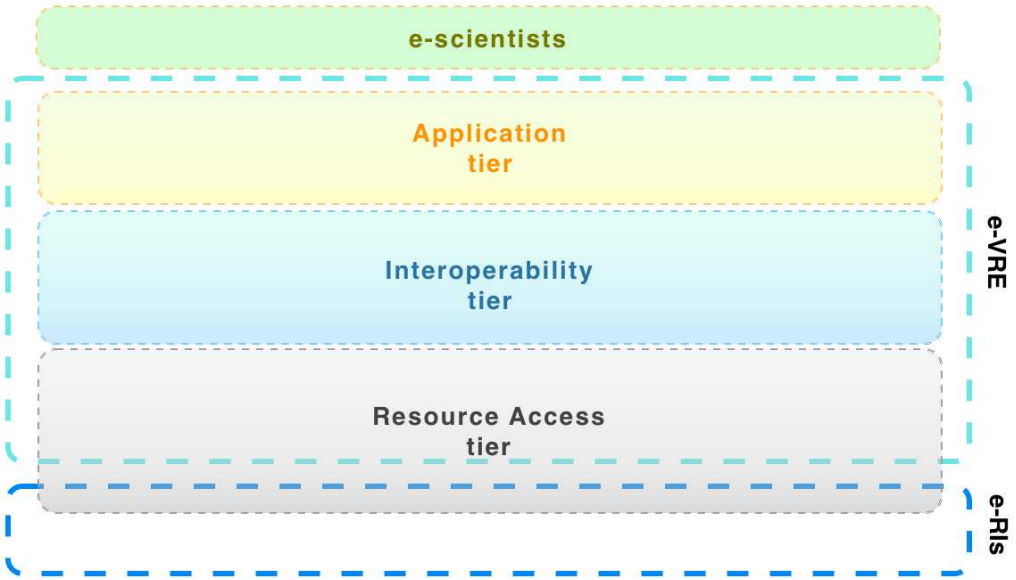


Figure 4 Architectural tiers in a VRE

Figure 4 depicts the logical tiers of e-VRE and shows their placement in an ideal space between the e-scientists that use the e-VRE and the e-RIs that provide the basic resources to the e-VRE.

Generally speaking a VRE system can be viewed as a dynamic framework; it "is the result of joining together new and existing components to support as much of the research process as appropriate for any given activity or role"[2]. To implement this fundamental non-functional requirement the e-VRE system has been designed following a *component-oriented* approach.

According to this approach a system is composed by an integration infrastructure where a set of software components can be deployed, these components implement the system functionalities and potentially can be specified, developed and deployed independently of one another.

Based on these considerations and on the analysis of the requirements, for the basic integration infrastructure of e-VRE we have individuated a set of basic functionalities grouped into six *conceptual components*:

- The e-VRE management is implemented in the **System Manager** component. The System Manager can be viewed as the component enabling Users to use the *core* functionalities of the e-VRE: access, create and manage resource descriptions, query the e-VRE information space, configure the e-VRE, plug and deploy new tools in the e-VRE and more.
- The **Workflow Manager** enables users to create, execute and store business processes and scientific workflows.

[2]Fraser M. "Virtual Research Environments: Overview and Activity" 30-July-2005, http://www.ariadne.ac.uk/issue44/fraser/

- The **Linked Data (LD) Manager** is the component that uses the LOD (Linked Open Data) paradigm, based on the RDF (Resource Description Framework) data model, to publish the e-VRE information space - i.e. the metadata concerning the e-VRE and the e-RIs in a form suitable for end-user browsing in a SM (Semantic Web)-enabled ecosystem.
- The **Metadata Manager** (MM) is the component responsible for storing and managing resource catalogues, user profiles, provenance information, preservation metadata used by all the components using extended entity-relational conceptual and object-relational logical representation for efficiency.
- The **Interoperability Manager** provides functionalities to implement interactions with e-RIs resources in a transparent way. It can be viewed as the interface of e-VRE towards e-RIs. It implements services and algorithms to enable e-VRE to: communicate synchronously or asynchronously with e-RIs resources, query the e-RIs catalogues and storages, map the data models.
- The **Authentication, Authorization, Accounting Infrastructure (AAAI**) component is the responsible for managing the security issues of the e-VRE system. It provides user authentication for the VRE and connected e-RIs, authorisation and accounting services, and data encryption layers for components that are accessible over potentially insecure networks.

Figure 5 shows how these six components are distributed on the 3-tier space introduced above.

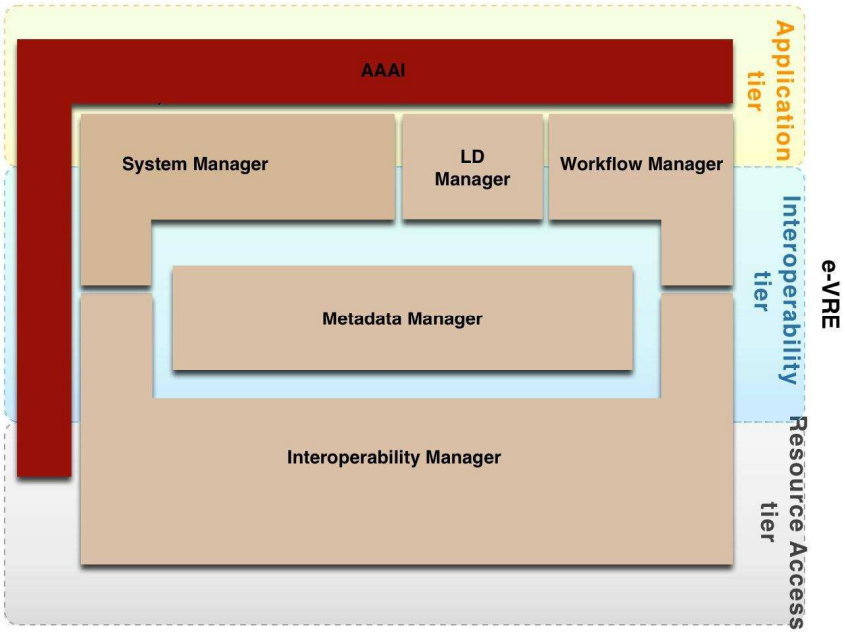


Figure 5 Conceptual components and logical tiers

4.2 Component Overview

Every conceptual component will be realised by one or more actual software components and possibly sub-components. The list of these components and subcomponents is given in the following Table, also given as Table 1 of the Annexes. All components named in the Table will be specified in the next Section.

Component	Sub-components	Description
AAAI Component		Manages security, privacy and trust aspects of the e-VRE and its connections to the e-RIs
	Authentication	Manages user authentication for the e-VRE and connected e-RIs (single sign on), interfaces with external identity provider services.
	Authorization	Manages user authorisations (role based access) based on (CERIF) metadata provided by the Metadata Manager.
	Accounting	Manages accounting and billing of resources for which payment is required, based on (CERIF) metadata provided by the Metadata Manager.
	Encryption	Provides encryption facilities.
Metadata Manager (MM)		Manages metadata about e-VRE entities: resource descriptions, user descriptions, provenance information, preservation metadata etc. (CERIF)
	User Catalogue	Contains user profiles and preferences
	Resource Catalogue	Contains metadata about resources available in e-VRE, i.e. datasets, services, workflows, instruments, networks of sensors, software applications etc
	Preservation Catalogue	Contains information related to the preservation process
	Provenance Catalogue	Contains metadata related to provenance
Interoperability Manager (IM)		Manages interactions with e-RIs
	Query Manager (QM)	Manages local and distributed queries, collects result sets
	Data Model Mapper (DMM)	Manages data and query format conversion
	Adapters	Components that synchronously interact with e-RIs resources

	Message-Oriented Middleware (MOM)	Manages asynchronous interactions with eRIs resources using messaging protocols
	e-VRE Web Services (e-VRE WS)	Enable external applications to interact with e-VRE
Workflow Manager (WM)		Manages business processes and scientific workflows, using the Metadata Manager for storing information on workflows
	Workflow configurator	Provides functionalities to build/edit/store *execution plans,* to control and monitor processing flows execution.
	Workflow executor	Manages workflow execution, including data staging
	Workflow repository	Provide functionalities to store and retrieve workflows, workflows will be published using LD manager
Linked Data Manager (LDM)		Manages the publication of information in e-VRE as Linked Open Data
	SPARQL Endpoint	Allows retrieving resources and services published by e-VRE as RDF documents
	LD API	The LD API maps CERIF metadata records in RDF, implements metadata enrichment of RDF records, i.e. adds to records typed links to vocabularies and thesaurus entries, Implements content negotiation
System Manager (SM)		Implements functionalities to *define* and manage the VRE, e.g. specify the resources, specify the apps, and to deploy the defined VRE in the available resources.
	Node Manager (NM)	Implements the functionalities to deploy, manage and run an instance of e-VRE on a specific hardware
	User Manager (UM)	Manages user profiles and provides collaboration/ communication functionalities for users. It provides the functionalities to add/update/remove user profiles, to set up users permissions, to manage users preferences, to configure users working environments
	Resource manager (RM)	Manages resource information implementing add/update/remove operations on resource descriptions, associating resources to security policies, etc.
	App Manager (AM)	Provides functionalities to deploy and manage applications that operate on e-VRE resources. It can be used also to embed applications such as Wiki or forums etc.

4.3 Components in detail

This Section describes each component introduced in the previous Section, providing:

- the methods in each interface of the component, along with the signature of each method
- the sub-components of the components, illustrated in a component diagram showing, as before, the interfaces provided and used by each sub-component.
- Interfaces to systems outside of VRE4EIC: e-Research Infrastructures (e-RIs) and e-Infrastructures (e-Is)

These specifications do not define Types in details; this is postponed until the final architecture will be released.

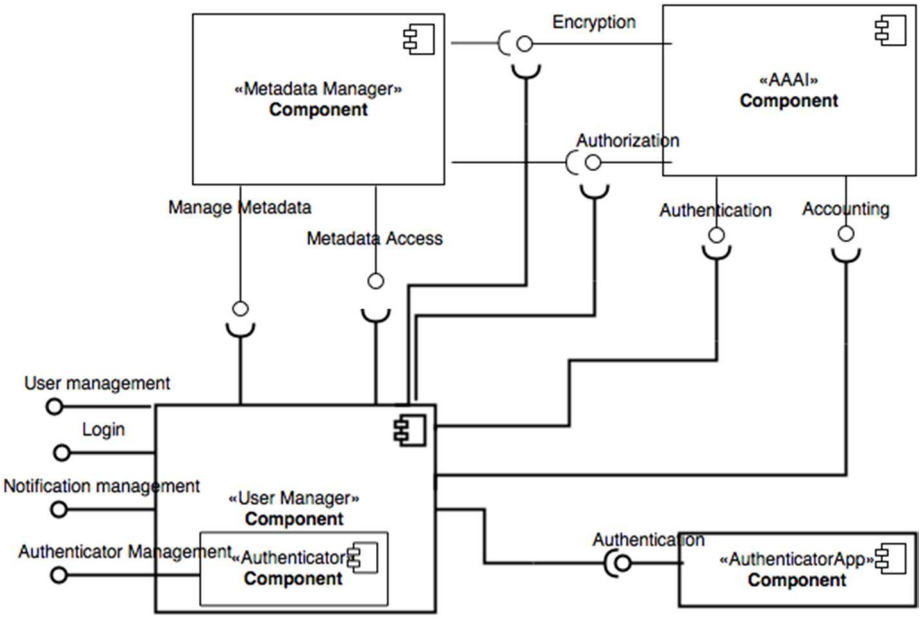
As general principle, CERIF entities will be used to model types when possible. For instance: UserCredentials, UserProfile, and UserQuery will be modeled starting from the corresponding CERIF entities. Appropriate classes will be created for the types that cannot be modeled by CERIF entities.

4.3.1 User Manager component

The User Manager is the component responsible for managing User Profiles, providing Authentication mechanisms and enabling users to receive Notifications for events they have subscribed.

To perform its activities the User Manager interacts with:

- Metadata Manager to store/retrieve/update User Profiles and Subscriptions/Notifications
- AAAI to implement authentication, encryption, authorization and accounting tasks
- AuthenticatorApp to implement login via external authenticator



The User Manager component provides four interfaces. Each such interface is described in a separate paragraph below, including a table providing the methods of the interface. Each method is described by naming the operation, the parameter list, and the return type of each method. In addition, the CERIF entity involved in the method is provided, if any.

4.3.1.1 User Manager: *User management* interface description

Operation	Parameter-list	Return type	Map to CERIF entity	Notes
createUserProfile	user:UserProfile	ReturnValue	UserProfile	Creates a user profile
updateUserProfile	userId:String, user:UserProfile	ReturnValue	UserProfile	Updates the profile userId
removeUserProfile	userId:String	ReturnValue		Deletes the profile userId
getUserProfile	userId:String	UserProfile	UserProfile	Retrieves the profile userId
getUserProfile	creds:UserCredentials	UserProfile	UserProfile	Gets the profile with provided credentials
getUserProfile	query:UserQuery	UserProfile[0..*]	UserProfile	Gets a list of profiles (wildcards allowed in query).

4.3.1.2 User Manager: *Notification management* interface description

Events related to RI resources are captured via Resource Manager and registered in Metadata Manager. The User Manager uses the Metadata Manager to check status changes.

Operation	Parameter-list	Return type	Map to CERIF entity	Notes
subscribeEvent	userId:String, events:EVREEvent[0..*]	ReturnValue	EVREEvent	Subscribes to a list of events
checkEvent	userId:String, eventId:String	ReturnValue		Returns the status of the specific event
checkEvents	userId:String	ReturnValue[0..*]		Returns the status of all subscribed events
getSubscribedEvents	userId:String	EVREEvent[0..*]	EVREEvent	Returns list of events subscribed by userId

4.3.1.3 User Manager: *login* interface description

The idea is that VRE4EIC authentication services could be based on scoped credentials assigned to a User or an Entity and controlled by authenticators (https://en.wikipedia.org/wiki/Authenticator).

Please note that we are talking of the authentication process between an eVRE user and the eVRE system, it will 'wrap' the protocols adopted by VRE4EIC AAAI infrastructure.

The scoping of the credentials must be enforced jointly by a User Agent implementing the VRE4EIC authentication API and an authenticator that holds the credential, by constraining the availability and usage of credentials.

Scoped credentials must be located on authenticators, which can use them to perform operations subject to user consent.

According to outcome of D2.1 we should have two types of authenticator:

- Authenticators located in the same device (e.g., smart phone, tablet, desktop PC) as the user agent is running on.
- Authenticators operate autonomously from the device running the user agent, and accessed via network or other protocols. This last part is needed mainly to implement requirements about external instruments or devices (DRQ2, DRQ4, DRQ5…)

To implement this behaviour we designed three interfaces: the *login* and the *Authenticator Management* interfaces provided by the User Manager component and the *Authentication* interface provided by a component called AuthenticatorApp that implements the client side functionalities of the authentication mechanism.

Operation	**Parameter-list**	**Return type**	**Map to CERIF entity**	**Notes**
login	creds:UserCredentials	ReturnValue		
login	authenticatorId:String, deviceId:string	ReturnValue		Uses external Authenticator (see: UC 21) for login
logout	userToken:String	ReturnValue		Signs out the user

4.3.1.4 User Manager: *Authenticator Management* interface description

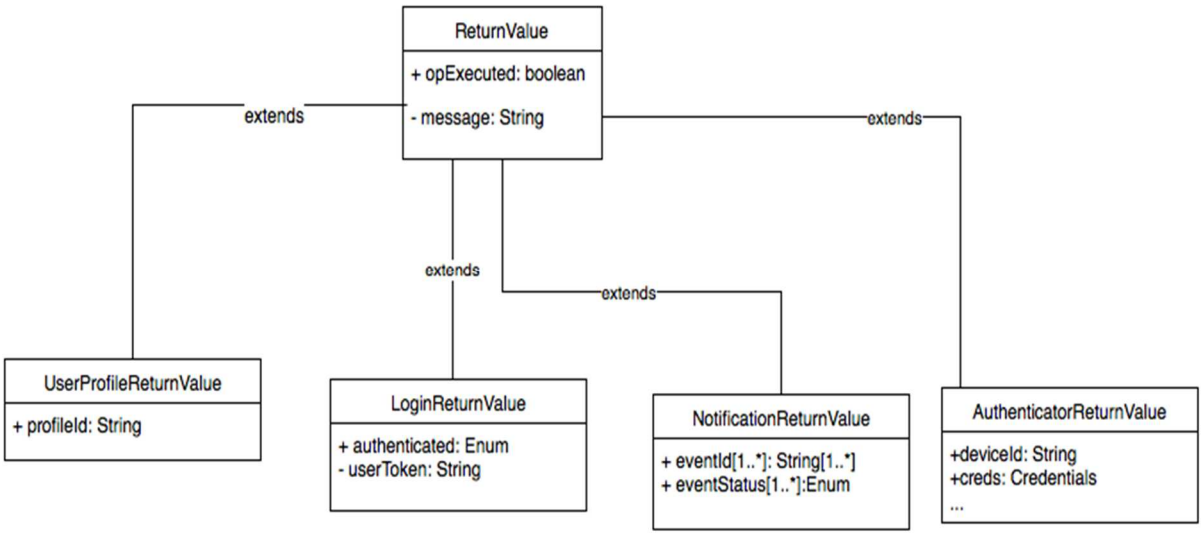
Operation	Parameter-list	Return type	Map to CERIF entity	Notes
registerAuthenticator	creds:UserCredentials, authenticatorId:String	ReturnValue		Register external authenticator
removeAuthenticator	creds:UserCredentials, authenticatorId:String	ReturnValue		Remove external authenticator

4.3.2 AuthenticatorApp: *Authentication* interface description

Operation	Parameter-list	Return type	Map to CERIF entity	Notes
authenticate	reqInfo:info	ReturnValue		The reqInfo contains information about device requesting authentication and the eVRE service that has been requested.
synchCredentials	authenticatorId:String, creds:UserCredentials[1..*]	ReturnValue		Synchronize credentials with the eVRE

Return Value

The class diagram below shows the topmost levels of a type hierarchy of return values.

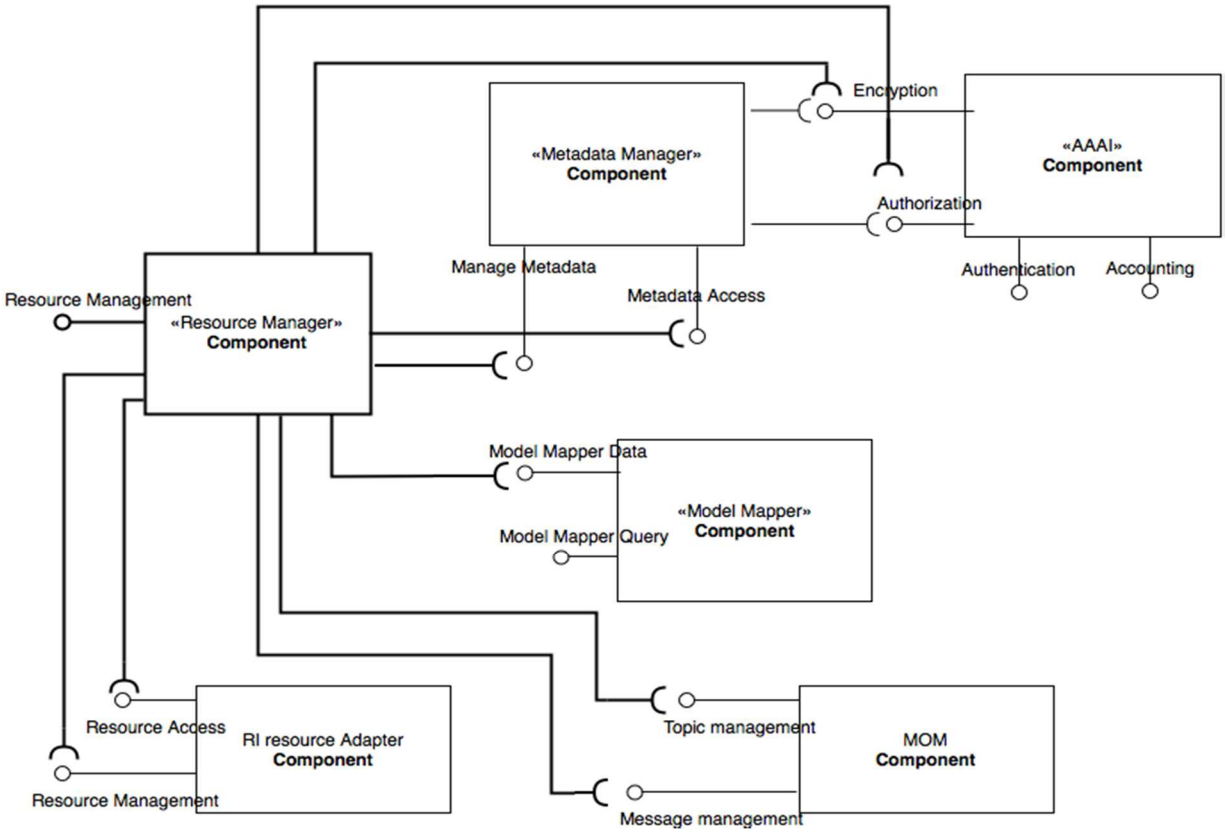


In the diagram, the main ReturnValue type returns a message and a Boolean reporting if the operation has been executed correctly or not. It is extended by four *subtypes* reporting the relevant information that for a specific operation, for instance the LoginReturnValue type returns also the token identifying the user and the result of the authentication.

4.3.3 Resource Manager component

The Resource Manager is the component responsible for managing information about resources provided by RIs and other infrastructures, it communicates with remote resources via Adapters or asynchronous messaging. The Resource Manager interacts with:

- Metadata Manager: to store, manage and retrieve information about resources
- AAAI component: to check permissions when interacting with external resources and to use encryption functionalities if needed
- Model Mapper Component: to map data when interacting with external resources
- RI Resource Adapter: for synchronous interactions with the external resource provided by a RI
- MOM component for asynchronous interactions with the external resource



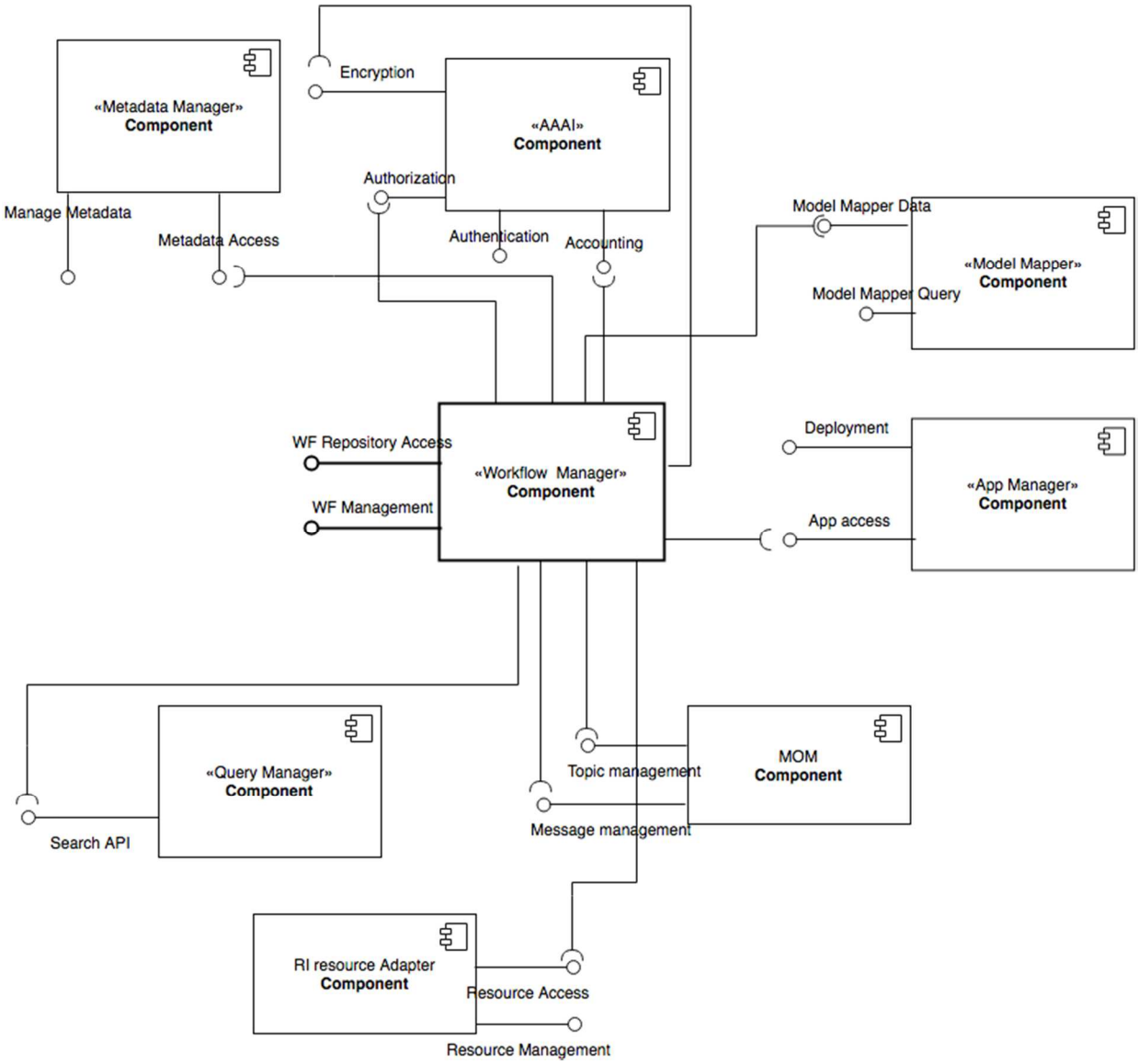
4.3.3.1 Resource Manager: *Resource management* interface description

Operation	Parameter-list	Return type	Map to CERIF entity	Notes
addResourceProfile	resource:ResProfile	ReturnValue	ResProfile	
updateResourceProfile	resourceId:String, resource:ResProfile	ReturnValue	ResProfile	
removeResourceProfile	resourceId:String	ReturnValue		
getResourceProfiles	query:ResQuery	ResourceProfile[0..*]	ResProfile	Wildcards can be specified in the query
getResourceProfile	resourceId:String	ResourceProfile	ResProfile	
isResourceAvailable	resourceId:String	ReturnValue		Checks if the resource is currently available in the RI, if RI provides this service. Depending on the RI service the return value can include info like: ETA for downtime, current use rate etc

4.3.4 Workflow Manager Component

The Workflow Manager is responsible for managing both Business and Scientific workflows[3].

In our vision Scientific Workflows represent experiments conducted by scientists, therefore the WF component will provide a workflow repository and interoperate with other workflow repositories to facilitate the reuse and reproducibility of scientific experiments. Information about workflows are stored in the Metadata Manager and will be published also as Linked Open Data via LD Manager.



[3] Bertram Ludäscher, Mathias Weske, Timothy McPhillips, and Shawn Bowers. Scientific workflows: Business as usual?,7th Intl. Conf. on Business Process Management (BPM), LNCS 5701, Ulm, Germany, 2009

4.3.4.1 Workflow Manager: Wf Repository Access interface

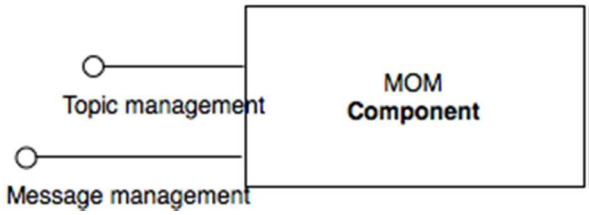
Operation	Parameter-list	Return type	Map to CERIF entity	Notes
getWF	idWF:String	WorkFlowDescription	WorkFlowDescription	
getWFs	idWFs:String[1..*]	WorkFlowDescription[0..*]	WorkFlowDescription	
getWFs	wfQuery	WorkFlowDescription[0..*]	WorkFlowDescription	Wildcards allowed

4.3.4.2 Workflow Manager: Wf management interface

Operation	Parameter-list	Return type	Map to CERIF entity	Notes
createWF	wf: WorkFlowDescription	ReturnValue	WorkFlowDescription	
updateWF	idWF:String	ReturnValue		
deleteWF	idWF:String	ReturnValue		
getWFs	idWFs:String[1..*]	WorkFlowDescription [0..*]	WorkFlowDescription	
executeWF	idWF:String	ReturnValue		
stopWF	idWF:String	ReturnValue		
pauseWF	idWF:String	ReturnValue		
getWFStatus	idWF:String	ReturnValue		

4.3.5 Message Oriented Manager (MOM) component

The MOM component is responsible for implementing asynchronous interactions with eRIs resources. It implements the interactions supporting messages exchanging between eRIs and eVRE components.



4.3.5.1 MOM: Topic interface

Operation	Parameter-list	Return type	Map to CERIF entity	Notes
getTopics	query:TopicQuery	Topic[0..*]	Topic	Wildcards allowed
createTopic	topic:Topic	ReturnValue	Topic	
updateTopic	topicId:String	ReturnValue		
removeTopic	topicId:String	ReturnValue		

4.3.5.2 MOM: Message Management interface

Operation	Parameter-list	Return type	Map to CERIF entity	Notes
addMessage	msgs:Message[1..*]	Return values		Routing info stored in messages
getMessages	topicIds:String[1..*]	Message[0..*]	Message	
getMessages	topicId:String[1..*], query:MessageQuery	Message[0..*]	Message	Wildcards allowed
getTopics	query:TopicQuery	Topic[0..*]	Topic	Wildcards allowed
subscribeTopic	topicIds:[1..*]	ReturnValue		
removeSubscription	topicIds:[1..*]	ReturnValue		
checkTopic	topicIds:[1..*]	ReturnValue		

4.3.6 Resource Adapter

The Resource Adapter indicates a set of components: an Adapter is a software module that wraps an external eRI resource; essentially Adapters reduce dependency of eVRE from eRI resources.

The Adapter acts as a middleware between an eVRE component and an eRI resource, it:

- Receives requests from the eVRE component and executes them by calling a service managing the resource
- Waits for the answer
- Returns the answer to the eVRE component

Adapters are specific for eRI resources and use synchronous standard protocols to interact with the resource. The adapter could be deployed in the eVRE system or in the eRI environment; it could also be split in subcomponents.

Interfaces of Adapters depend on the resource they wraps.

4.3.7 APP Manager Component

The goal of this component is to provide functionalities to enable external applications to be embedded and used into the E-VRE system. Generally speaking this means that such a component should implement a set of facilities to manage:

1. the deployment of external applications in E-VRE
2. the life-cycle of these applications (install/start/stop/update/uninstall)
3. the publication and the discovery of these applications

The idea is to built the App Manager as a lightweight, unobtrusive component that will interact with external applications to track their lifecycle and their usage.

The App Manager is a crucial component for E-VRE, we plan to build it as a lightweight, unobtrusive software module that interacts with external applications to track their lifecycle and their usage.

However, creating an App Manager able to automatically manage lifecycle and usage for every possible external application embedded in the EVRE is not feasible: we'll individuate a set of standards and technologies and implement the App Manager for applications adopting those standards (for instance Servlet level 3 specification is a good candidate), applications not implementing the selected technologies will be embedded by extending the App manager with *ad hoc* sub components.

4.3.8 Metadata Manager (MM) Component

The Metadata Manager is responsible for storing, manipulating and exposing metadata information about various resources. It contains a set of catalogues and repositories and stores information with respect to a set of predefined schemas.

The Metadata Manager component contains a set of subcomponents that deal with particular functionalities: the Thesaurus, the Provenance Manager, the Preservation Manager etc.

The Metadata Manager exposes its contents through the **GetMetadata** interface. The QueryManager for instance uses this particular interface when searching for particular resources.

4.3.8.1 Metadata Manager: GetMetadata interface

Operation Name	**Parameter-list**	**Return type**	**Map to CERIF entity**	**Notes**
getResourceMetadata	String: resourceURI Collection<String> graphspaces	Collection<Triple> results		The method takes as input the URI of a resource, and the corresponding graphspaces and returns the contents of the metadata catalogue as a collection of triples (i.e. <S, P, O>, where S=subject, P=predicate and O=Object).
getResourceMetadataUsingType	String: resourceURI Collection<String> graphspaces Enum: metadataType	Collection<Triple> results		The method is similar in spirit with *getResourceMetadata* method, however it also contains a type parameter to specify the exact metadata type that is requested (i.e.. owner of a resource).
searchForMetadata	Collection<String> queryTerms Collection<String> graphspaces	Collection<Triple> results		The method takes as input a set of queryTerms and the graphspaces to search for and searches in the metadata catalogues for these terms. Finally it returns the results as a collection of triples.

The Metadata Manager also contains the ManageMetadata interface that allows users to add new information or to update existing information from the metadata catalogues. This interface is exploited from almost all the sub-components of the SystemManager (ResourceManager, UserManager, etc.). These components invoke the appropriate methods of the interface every time some entry needs to be updated; the rationale is that each Manager (from the System Manager) will update the appropriate metadata catalogue (i.e., the User Manager will be able to update only the metadata about users). However, in some cases the ManageMetadata can be used by particular users or agents (having an administrative role) for updating information in the metadata catalogues.

Furthermore, the LDManager uses this interface for adding metadata information as regards to the publishing of Data also as Linked Open Data.

4.3.8.2 Metadata Manager: ManageMetadata interface

Operation Name	Parameter-list	Return type	Map to CERIF entity	Notes
insertUpdateMetadata	String: resourceURI Enum: metadataType String: metadataValue String: graphspace	void		The method updates (or adds if such information does not exist) particular information about the metadata of a resource. The type of the metadata and the corresponding value are also given in the parameters list. The new triples is being added under the given graphspace.
deleteMetadata	String: resourceURI String: graphspace	void		This methods deletes information about a specific metadata resource from the given graphspace.

Apart from the interfaces of the Metadata Manager component, there is also the ThesaurusAPI interface, offered by the Thesaurus subcomponent, which is being used by the Query Manager and the UI components to retrieve suggested terms during search and spell checking. In addition, the ProvenanceAPI interface, offered by the Provenance Manager subcomponent, enables the storing of information about Queries and workflows and is being used by the Workflow Manager and the Query Manager.

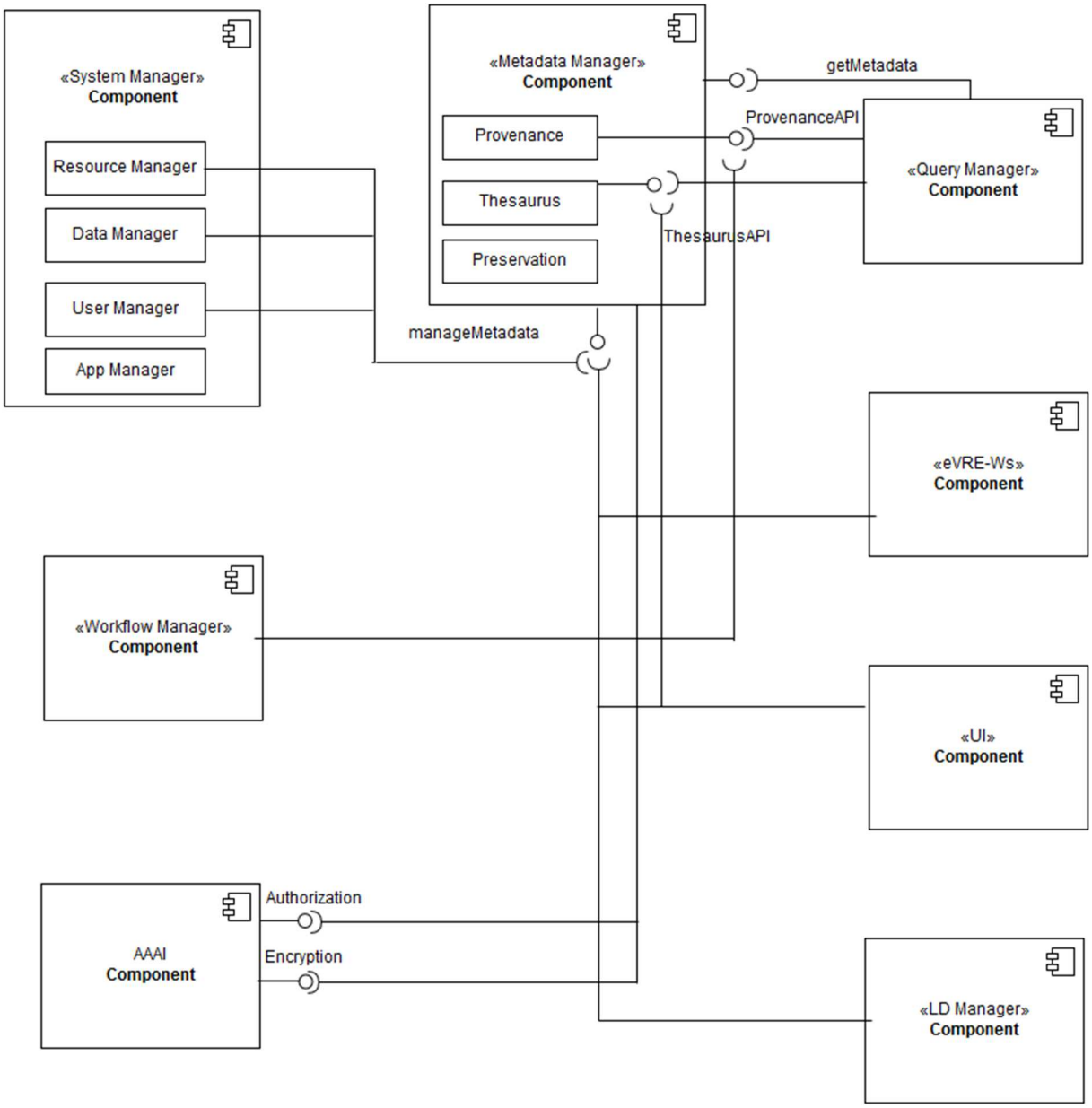
4.3.8.3 Thesaurus Component: ThesaurusAPI interface

Operation Name	Parameter-list	Return type	Map to CERIF entity	Notes
getSuggestedTerms	String: queryTerm	Collection<String> suggestedTerms		The method takes as input a queryTerm, and searches in the catalogue maintained by the Thesaurus component for suggested terms. For instance if the queryTerm is “inf” then it will return (among others) the terms “infrastructure”, “inference”, etc.
getSimilarTerms	String: queryTerm int: editDistance	Collection<String> similarTerms		The method takes as input a queryTerm and a value for the editDistance function to perform spell checking. It returns a ranked collection of potential corrections for the given query term.

4.3.8.4 Provenance Component: ProvenanceAPI interface

Operation Name	Parameter-list	Return type	Map to CERIF entity	Notes
getProvenanceMetadata	String: resourceURI	Collection<Triple> results		The method takes as input the URI of a resource and returns all the available provenance metadata of the resource as a collection of triples.
getProvenanceMetadataUsingType	String resourceURI Enum: metadataType	Collection<Triple> results		The method is similar in spirit with *getProvenanceMetadata*, however the current one uses one more parameter for defining the type of the metadata that is requested, and returns the facts about provenance of the given resource as a collection of triples.
updateProvenanceMetadata	String: resourceURI Enum: metadataType String: metadataValue	void		The method updates (or adds if such information does not exist particular information about the provenance of a resource. The type of the metadata and the corresponding value are also given in the parameters list.

We should note that there is no direct connection between the Mapping Manager (subcomponent of the Model Mapper) and the Metadata Manager. The reason is that the Mapping Manager (as it is shown in the corresponding diagram) is being exploited by the Query Manager directly whenever it is required to perform mappings (over data or over a query).


«System Manager»
Component
Resource Manager
Data Manager
User Manager
App Manager
«Metadata Manager»
Component
Provenance
Thesaurus
Preservation
getMetadata
ProvenanceAPI
ThesaurusAPI
«Query Manager»
Component
manageMetadata
«eVRE-Ws»
Component
«Workflow Manager»
Component
«UI»
Component
AAAI
Component
Authorization
Encryption
«LD Manager»
Component

4.3.9 The Query Manager (QM) Component

The Query Manager is the component responsible for managing the querying capabilities of the infrastructure. This component receives as input a set of query requirements (in terms of keywords, query preferences, dataset catalogues, etc.) and manages the entire process of

- preparing the query,
- splitting it into subqueries,
- submitting the subqueries into the proper systems,
- receiving the results, and
- integrating them in order to send them to the user as a unified set.

The QueryManager is a core component (a super-component) that aggregates and exposes the functionalities of various subcomponents (i.e., Query Analyzer, Query Mediator, Query Integrator, Query Publisher, etc.).

The QueryManager exposes its functionalities through the SearchAPI interface. This interface contains the appropriate methods that take as input the query requirements (i.e., query terms, dataset catalogues to be searched, query preferences, etc.) given by a user/agent and, after performing all the query-related functionalities, it returns the results back to the user/agent. The SearchAPI can therefore be used from the eVRE WS and UI components (we should also stress that the latter components are responsible for validating that the user/agent has the rights to submit a query).

After receiving a request (through the SearchAPI interface), the QM determines that the query should split into subqueries with respect to the schema of the target catalogues. For this reason, it communicates with the ModelMapperQuery interface of the ModelMapper component, for translating the query into the appropriate subquery with respect to the target schema of the corresponding catalogue.

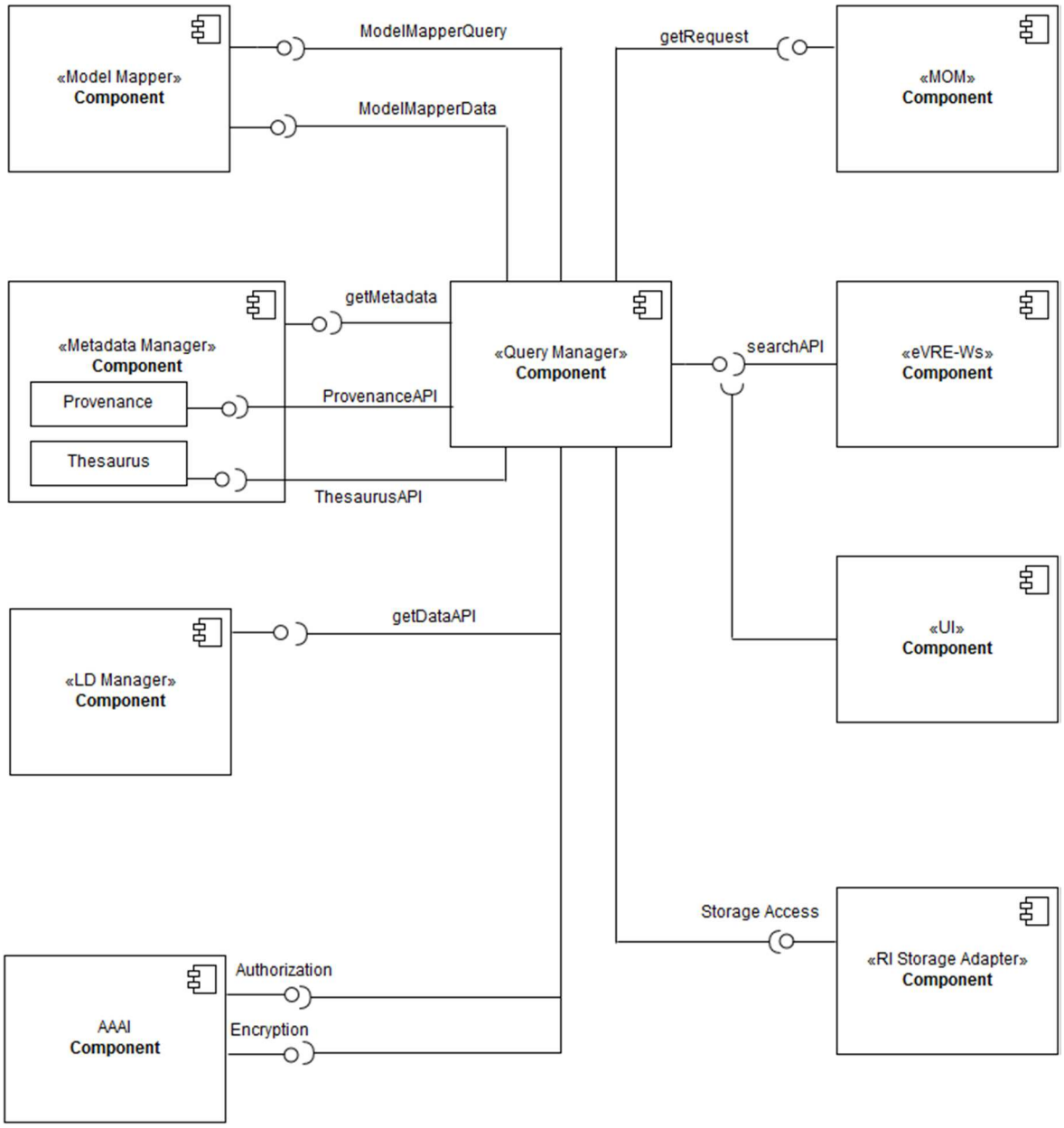
Afterwards, the appropriate subqueries are sent to the corresponding external resources through the GetRequest API of the MessageOriented component. After retrieving the results from the various resources, the QM exploits the ModelMapperData interface of the ModelMapper component, in order to transform them with the proper mappings into a common format.

Apart from the above, the QM can also search for data that has been published on the infrastructure as LinkedData through the **GetDataAPI** of the LDManager, and for data that exist in the metadata catalogue through the **getMetadata** interface of the **MetadataManager** component.

Finally, we should note that the QM also communicates with the **ThesaurusAPI** and **ProvenanceAPI** of the corresponding subcomponents of the **MetadataManager** for performing automatic corrections of keywords and recording the query evaluation provenance, respectively.

4.3.9.1 Query Manager Component: SearchAPI interface

Operation Name	Parameter-list	Return type	Map to CERIF entity	Notes
searchSimple	Collection<String> queryTerms Collection<Pair<String,String>> preferences	Collection<Result> results		The method takes as input a set of query terms, and a set of preferences - expressed as key-value pairs (i.e. perform ranking, filtering, etc.)- and searches locally, and finally returns a collection of results.
searchSimpleWithinRange	Collection<String> queryTerms Collection<Pair<String,String>> preferences int: startOffset int: limit	Collection<Result> results		The functionality is similar in spirit with *searchSimple* method, with the only difference that it takes as input the startOffset and the upper limit, to support paging of results (i.e. starting from result 1 bring 100 results).
searchFederated	Collection<String> queryTerms Collection<String> dataSources Collection<Pair<String,String>> preferences	Collection<Result> results		The method takes as input a set of queryTerms, a set of data sources, and a set of preferences -expressed as key-value pairs- and performs a federated search over the given data sources, and finally returns a collection of results.
searchFederatedWithinRange	Collection<String> queryTerms Collection<String> dataSources Collection<Pair<String,String>> preferences int: startOffset int: limit	Collection<Result> results		The functionality is similar in spirit with *searchFederated* method, with the only difference that it takes as input the startOffset and the upper limit, to support paging of results (i.e. starting from result 1 bring 100 results).


«Model Mapper»
Component
ModelMapperQuery
ModelMapperData
getRequest
«MOM»
Component
«Metadata Manager»
Component
Provenance
Thesaurus
getMetadata
ProvenanceAPI
ThesaurusAPI
«Query Manager»
Component
searchAPI
«eVRE-Ws»
Component
«UI»
Component
«LD Manager»
Component
getDataAPI
Storage Access
«RI Storage Adapter»
Component
AAAI
Component
Authorization
Encryption

4.3.10 The Model Mapper Component

The Model Mapper component is responsible for performing the required mappings and storing the particular information (the mappings themselves) in its repository.

The component contains the following subcomponents; (a) the Mapping Manager, which is responsible for adding and manipulating the available mappings, (b) the Data Transformer that exploits the available mappings for transforming a data source and (c) the Query Translator for performing query translations.

The Model Mapper component exposes its functionalities as regards the translation of queries through the **ModelMapperQuery** interface. This interface is being used from the QueryManager component (whenever it is required to translate a query with respect to a different model/ schema/ format).

4.3.10.1 Model Mapper Component: ModelMapperQuery interface

Operation Name	Parameter-list	Return type	Map to CERIF entity	Notes
transformQueryExpression	String: initialQuery String: targetSchema	String: queryExpr		This method takes as input a query and a description of the target schema and is responsible for transforming it so that it can be submitted to the target system. The method returns the expression of the query with respect to the target format.

It also has a **ModelMapperData** interface that exposes the functionalities as regards the data transformation, which is being used from the QueryManager component (whenever it is required to transform some data - i.e. results- and deliver them in an homogeneous way to the users/agents), the WorkflowManager and the SystemManager components.

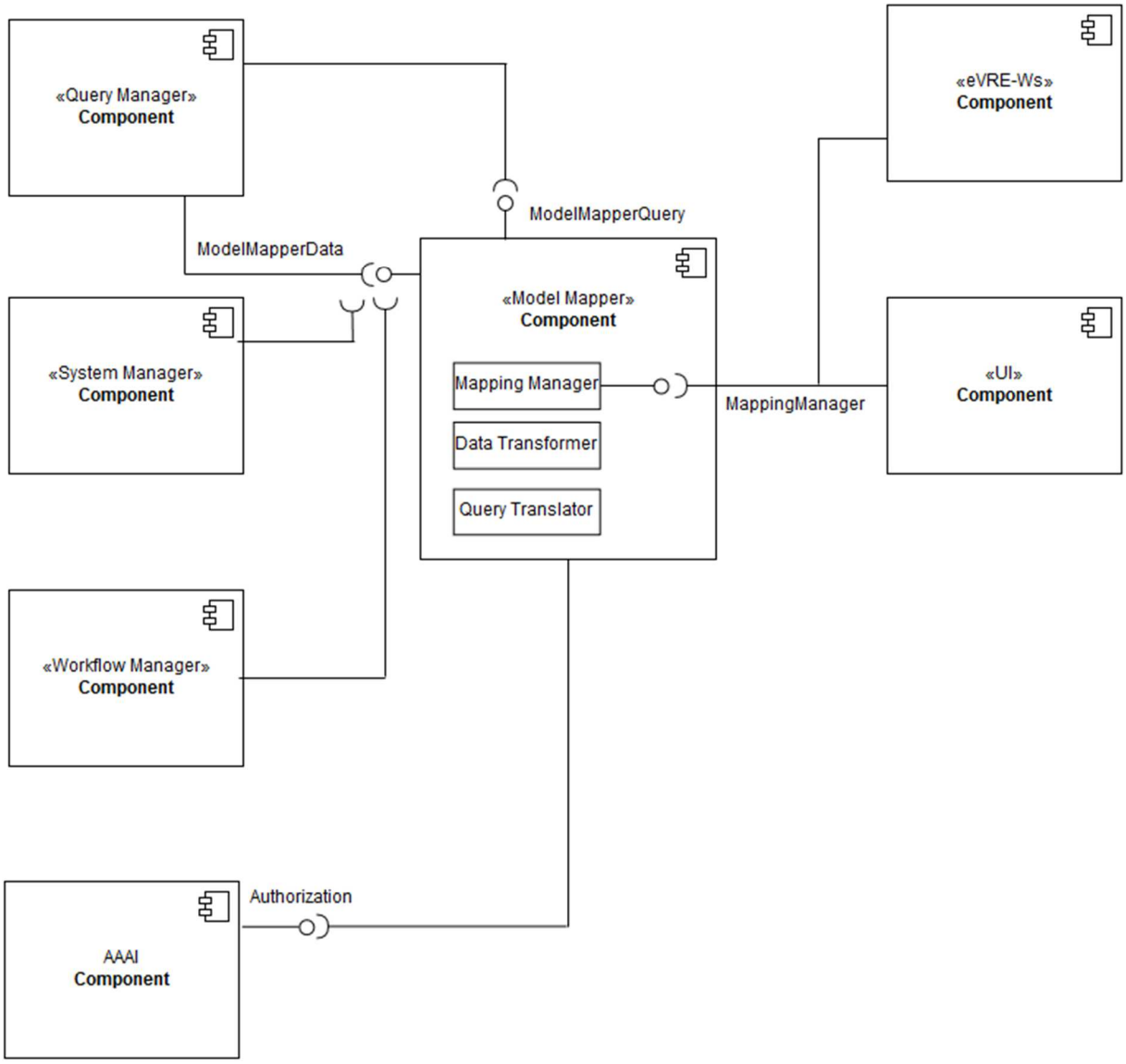
4.3.10.2 Model Mapper Component: ModelMapperData interface

Operation Name	Parameter-list	Return type	Map to CERIF entity	Notes
transformData	String: originalData String: targetSchema	String: transformedData		This method takes as input the textual description of some data a query and a description of the target schema and is responsible for transforming them with respect to the target schema. The method returns the textual description of the transformed data.

Finally the ModelMapper component has a MappingManager interface that contains all the functionalities as regards the manipulation of mappings.

4.3.10.3 Model Mapper Component: MappingManager interface

Operation Name	**Parameter-list**	**Return type**	**Map to CERIF entity**	**Notes**
getMapping	String: mappingID	String: mappingExpression		The method takes as input the ID of a mapping and returns the textual representation of the mappings (i.e. as an X3ML file)
addMapping	String: mappingID String: mappingExpression	void		The method takes as input a textual representation of a mapping (i.e. a X3ML [X3ML_Framework_IJDL_2016] file), and an ID and stores the mapping in the MappingManager catalogue.
updateMapping	String: mappingID String: mappingExpression	void		The method takes as input a textual representation of a mapping (i.e. a X3ML file), and an ID and updates an existing mapping in the MappingManager catalogue.
deleteMapping	String: mappingID	void		The method takes as input a textual representation of a mapping (i.e. a X3ML file), and removes the mapping from the MappingManager catalogue.



4.3.11 The LD Manager Component

The LD Manager is the component responsible for publishing resource descriptions with respect to the principles of Linked Open Data. The process of publishing contains the transformation of resources, the generation of (resolvable) URIs, their linking, etc.

Since the main functionality of the component is to support the publishing of metadata, it contains a PublishLDAPI interface. This interface is being exploited from the WorkflowManager, the eVRE WS and the UI components whenever it is requested to publish some data as Linked Open Data. In these cases in order to fetch the particular data that will be published it exploits the SearchAPI of the QueryManager component. In order to support the transformation of the given data the LDManager uses the ModelMapperData interface which is given from the ModelMapper component.

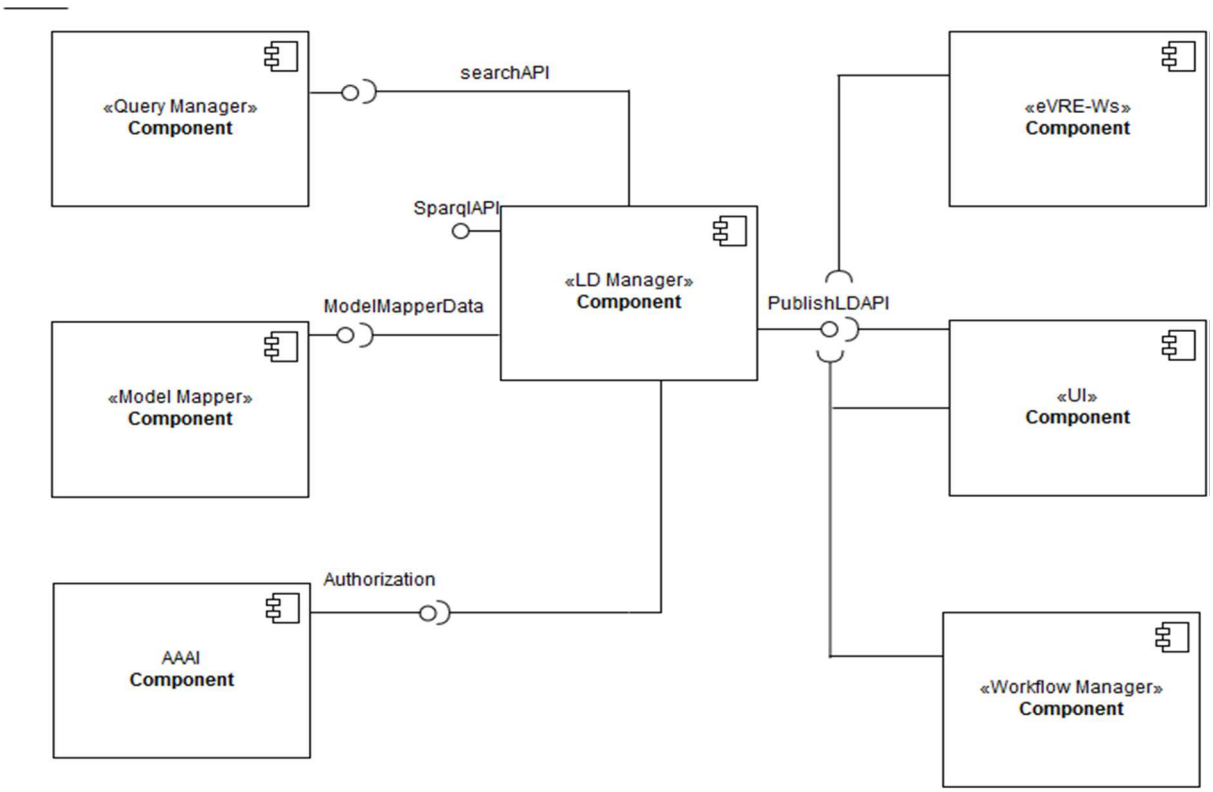
4.3.11.1 LDManager Component: PublishLDAPI interface

Operation Name	Parameter-list	Return type	Map to CERIF entity	Notes
publishLinkedData	String: publicationSourceURI	void		This method takes as input the publicationSourceURI in which the data will be published (i.e a named graph uri)

Moreover, the published data are exposed through the **SPARQL-API** interface.

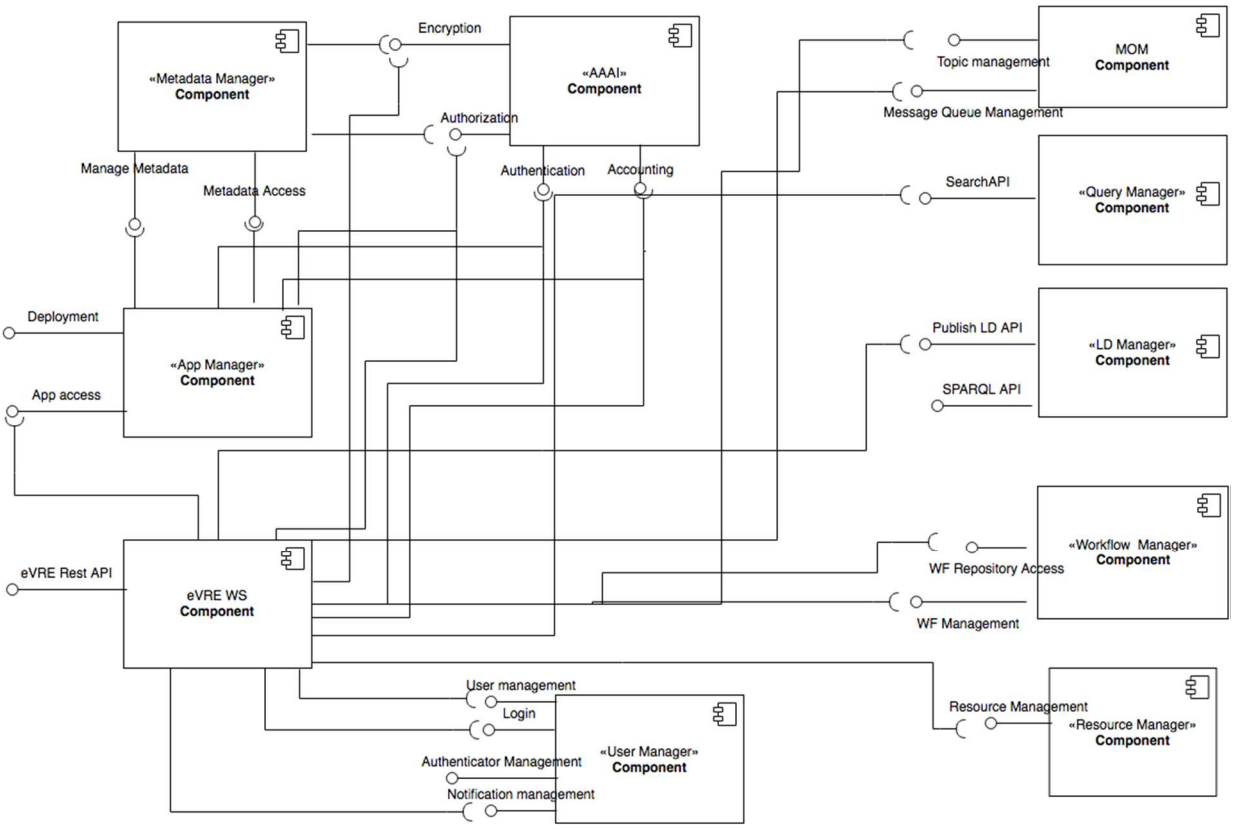
4.3.11.2 LDManager Component: SPARQL-API interface

Operation Name	Parameter-list	Return type	Map to CERIF entity	Notes
querySPARQL	String: sparqlQuery Format: returnType	String: sparqlResults		The method takes as input a SPARQL query, and the type of the returned results (i.e. XML, JSON, HTML, etc.), it executes the query and returns the results with respect to the given return type.



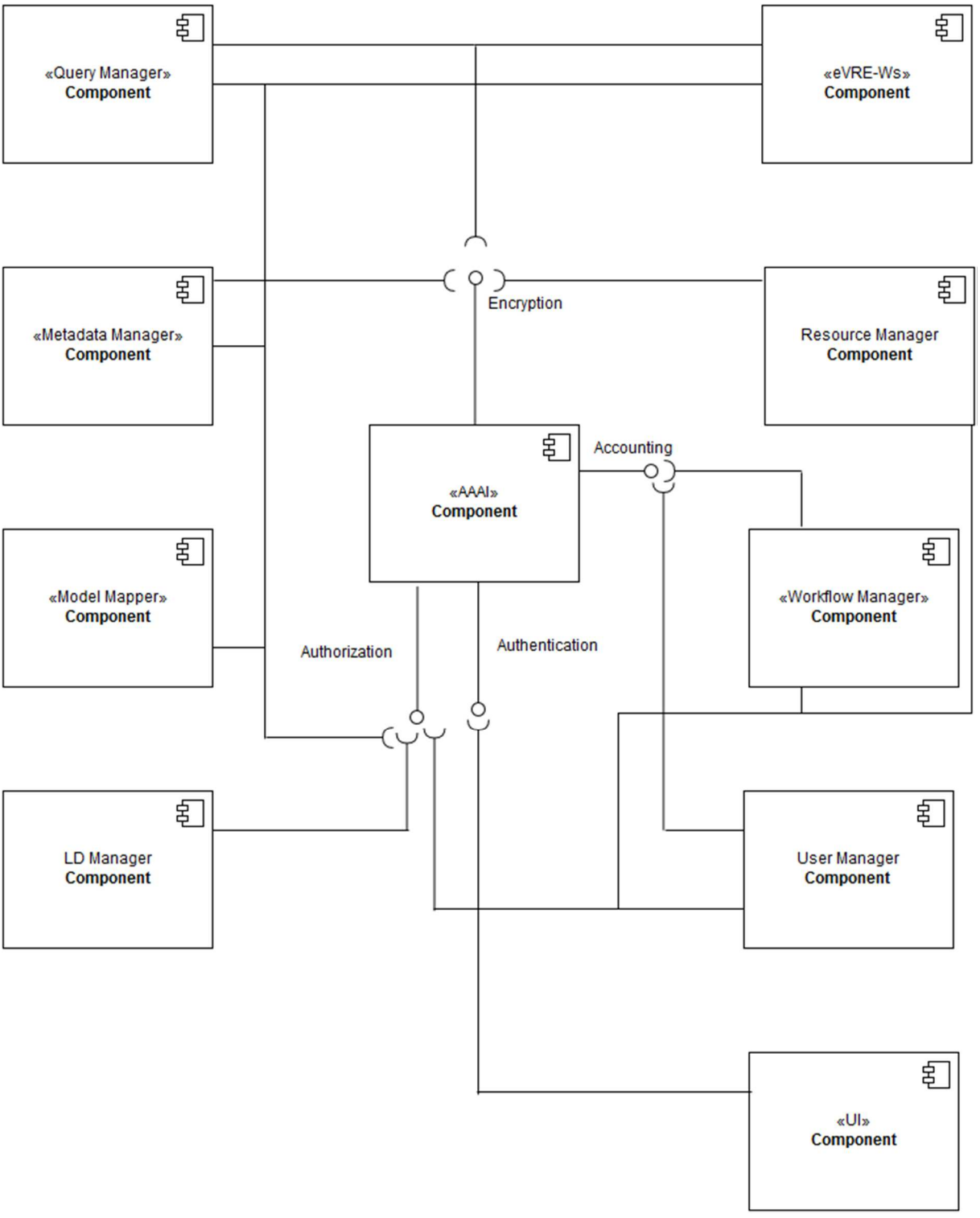
4.3.12 eVRE WS component

This component implements the Web Service API for eVRE architecture. It is a crucial component, it is used by external agents (applications) to access the functionalities of the eVRE. It could be also used as *service level integration middleware* for expanding the e-VRE with new functionalities.



4.3.13 AAAI Component

This component implements the authentication, authorisation, accounting and data encryption backend services for the eVRE architecture. It interfaces with external identity providers to enable single sign on across the various connected infrastructures. It delegates the login and other user interface services to the User Manager component described above. For any authenticated user, it provides authorization services by using attributes provided by the external identity provider (if any). These will be extended by using (CERIF) information by interfacing with the Metadata Manager component.



Operation Name	Parameter-list	Return type	Map to CERIF entity	Notes
authenticateUser	creds:UserCredentials	UserProfile	UserProfile	Delegate to federated identity service
authorizedUser	creds:UserCredentials resourceId:String, operationType:String	Boolean		Return true if user in her current role is authorized to perform given operation on given resource
billUser	creds:UserCredentials resourceId:String, amount:Number	Invoice		Bill user for using amount units of a given resource
encryptData	encryption:Scheme plainData	EncryptedData		Encrypt data using given scheme. Note that in the actual implementation, we expect this function to be embedded in the components that need them

4.4 References

[X3ML_Framework_IJDL_2016] Y. Marketakis, N. Minadakis, H. Kondylakis, K. Konsolaki, G. Samaritakis, M. Theodoridou, G. Flouris, M. Doerr. X3ML Mapping Framework for Information Integration in Cultural Heritage and beyond. International Journal on Digital Libraries, pp 1-19, Springer, DOI 10.1007/s00799-016-0179-1, May 2016.

5 Joining an eVRE instance

This Section presents a significant usage scenarios of the Reference Architecture, focussing on the operations that an e-RI must perform in order to join an existing eVRE as a participating e-RI.

The steps that the e-RI must consider planning and executing can be grouped in three main phases: *analysis*, *preparation* and *execution* phase. These three phases are ordered in time as shown in the

Figure below. Collectively they form a cycle that can be executed several times over time.

The *analysis* phase is mainly carried out by the management of the e-RI, and aims at establishing the objectives that the e-RI aims at achieving by joining the eVRE. More specifically, what are the resources, chiefly data and services, the e-RI wants to (1) access from the eVRE; and (2) share to the eVRE, that is made available to the eVRE. Concerning the first point, special consideration must be given to the resources that the e-RI wishes to have available from the eVRE, either internal, that is directly provided by the eVRE, or external, that is mediated by the eVRE. Concerning the second point, the e-RI must take into consideration that each resource it will make available to the eVRE will have to be: (a) endowed with a description that is as complete as possible with respect to the eVRE requirements, and of adequate quality, and (b) made accessible to the eVRE, which implies adding code, in the form of , *e.g.*, an adapter or a wrapper.

These considerations will allow the e-RI management to perform a cost/benefit analysis that would allow them to take an informed decision as whether or not to join the eVRE. After deciding to join, the e-RI enters into the preparation phase.

The *preparation* phase is carried out by the IT management of the e-RI, and its central task is to conduct a negotiation with the eVRE IT manager for agreeing on several important issues, such as:

- The e-RI's trust/security/privacy policies are compliant with those supported by the eVRE that it wishes to join. This includes the authentication, authorization and access interfaces that the eVRE requires from the e-RI.
- The descriptions of the resources that the joining e-RI wishes to share through the eVRE are rich enough for the Catalogue of the eVRE and automatically transformable into the eVRE Catalogue format.

- The alignment between the e-RI catalogue(s) and the eVRE catalogue is ensured by the e-RI according to the protocols and interfaces indicated by the eVRE.
- The resources that the joining e-RI wishes to share through the eVRE are accessible according to the protocols and interfaces indicated by the eVRE.

Once successfully concluded, the negotiation phase produces a plan of the actions that the e-RI must undertake in order to implement the negotiated agreements. This plan will be executed by the IT specialists in the execution phase.

The *execution* phase includes at least the following operations:

- The definition of the mappings from the e-RI catalog data model to the eVRE catalog data model
- The implementation of an aggregation infrastructure for the extracting, transforming (based on the previously defined mapping) and loading the e-RI resource descriptions from the e-RI Catalogue into the eVRE Catalogue, so that there is the best possible alignment between the two.
- The implementation of the resource access protocols and interfaces negotiated with the eVRE.

The cycle can be re-entered any time, from anyone of its phases, depending on the changes that occur in the e-RI. Policy changes may require to re-execute the analysis, leading to a new cost/benefit analysis with a possibly different outcome. Changes in the eVRE technological architecture may require a new preparation phase leading to a new plan. Changes in the e-RI technological architecture may require a new execution phase.

6 Assessment of the architecture

This Section assesses the Reference Architecture by relating the Reference Architecture and vision of e-VRE to the ongoing work in the area of VRE, and then by presenting sequence diagrams of the most important use cases, thereby showing the adequacy of the Reference Architecture from a functional point of view.

6.1 Introduction

In designing the e-VRE we have taken cognisance of past and ongoing work on:

1. VREs (also known as SGs (Science Gateways dominantly in North America) and as VLs (Virtual Laboratories, dominantly in Australasia);
2. e-RIs: e-Research infrastructures providing access to facilities and having assets of data, software, equipment, computing, users, publications etc.
3. e-Is: infrastructures which form the common set of utilities to be used by e-RIs; examples are GEANT, AARC2, EUDAT, EGI, PRACE, OpenAIRE.

The useful booklet produced by DG-CNECT on research infrastructures is a useful reference resource[4]. The VRE4EIC Project Proposal already classified the e-infrastructure scene into the components above.

6.1.1 VREs

There are 3 other H2020 RIA projects concerned with VREs: EVER-EST (geoscience); BlueBridge (blue, dominantly marine) and West-LIFE (Bio). These are at an early stage of development like VRE4EIC and so our considerations have relied on (a) information from the project websites (b) personal contacts especially with EVER-EST (where already joint meetings have taken place) and Blue Bridge (where the major partner is the same organisation (but a different group) as the architecture developers in VRE4EIC). There are significant differences in approach:

1. VRE4EIC is producing a reference architecture (and prototype demonstrator) that can bridge across e-RIs (and hence underlying e-Is) in a multidisciplinary manner; the other projects are restricted to particular domains;
2. BlueBridge produces a VRE with limited capabilities (compared with those of VRE4EIC in Table 2 of the DoA) and is tightly coupled to the underlying e-RIs;
3. EVER-EST is using research objects; this binds data and code in a particular way that restricts openness and interoperability, which is unacceptable for the objectives of VRE4EIC. Nonetheless we are working together and believe there are opportunities for co-development.
4. WEST-LIFE does not give much information on the web pages. It appears to be centered on EGI and presumably plans to use their existing technologies which are lower-level modularised components for self-assembly by the e-RI to form a VRE. Nonetheless we are trying to establish more detailed technical contacts.

[4] https://ec.europa.eu/digital-single- market/en/news/e-infrastructures- making-europe- best-place-research-and- innovation

Outside of these H2020 projects for VRE4EIC participants the scientific coordinator has initiated a RDA IG (research Data Alliance Interest Group) jointly with EVER-EST. This allows us to evaluate the work on SGs and VLs. Furthermore, the VRE4EIC scientific coordinator is on relevant program committees for workshops and conferences on SGs.

In general SGs are portals to datasets. Some have analytical, visualisation and simulation capabilities. Some provide access to –and steering of – equipment. Some provide access to specialist computing resources and some provide collaboration tools. However – in contrast to VRE4EIC - in general they are constructed on top of one or more – e-RIs in a particular domain.

VLs have taken a different approach. VLs are constructed from a pool of general software modules, available datasets and user groups. Again each VL tends to be domain specific and linked with one or a small number of e-RIs. Nonetheless VRE4EIC will continue to track the evolution of VREs, SGs and VLs and cooperate wherever possible to increase the potential of interoperability in an open research environment.

6.1.2 e-RIs

VRE4EIC has within the consortium representation from ENVRIplus and EPOS: two large e-RIs in the environmental and geoscience domains. These were chosen because of an intersection of personnel in the partners thus building on pre-existing relationships.

However, as indicated in the project proposal, the partners between them have some knowledge of the major EC-funded e-RIs, particularly those of ESFRI. In D2.1 we characterised e-RIs from multiple domains to ensure that we understood their capabilities and offerings. A key aspect of VRE4EIC is to produce an e-VRE which does not duplicate the functionality of the e-RIs but builds upon them, orchestrating and facilitating user access and utilisation of them - subject to rights, security, privacy and performance considerations.

In general the e-RIs provide portal access to discover and download assets such as data and software. Some provide workflow capabilities and access to computing resources. They provide access to facilities and equipment. Thus the e-VRE of VRE4EIC has to facilitate user access to the e-RIs, overcome the heterogeneities (interoperation) and provide the services identified in the user requirements (D2.1) that are not provided by the e-RIs as characterised (D2.1). This is the reason why D3.2 Gap Analysis – between D2.1 and D3.1 - is important to the developing VRE4EIC architecture for our e-VRE.

6.1.3 e-Is

Most of the e-RIs utilise the e-Is to provide required utility functions. The e-VRE of VRE4EIC will thus – in general – utilise the e-Is via the e-RIs. However, the e-VRE will itself require to use GEANT, an AAAI environment such as AARC2, some local / temporary storage and curation with provenance facilities (EUDAT), access to scholarly publications (OpenAIRE) and access to – and utilisation of – advanced computing facilities such as EGI, PRACE or EOSC.

6.2 Use Case Sequence Diagrams

In previous sections, with emphasis in Section 4, we highlighted the main characteristics of the Reference VRE Architecture and the needs that drive the proposed composition of components. From the functional point of view, these needs, which are the result of use cases and requirements obtained by communication with VRE users, should be reflected in the architecture.

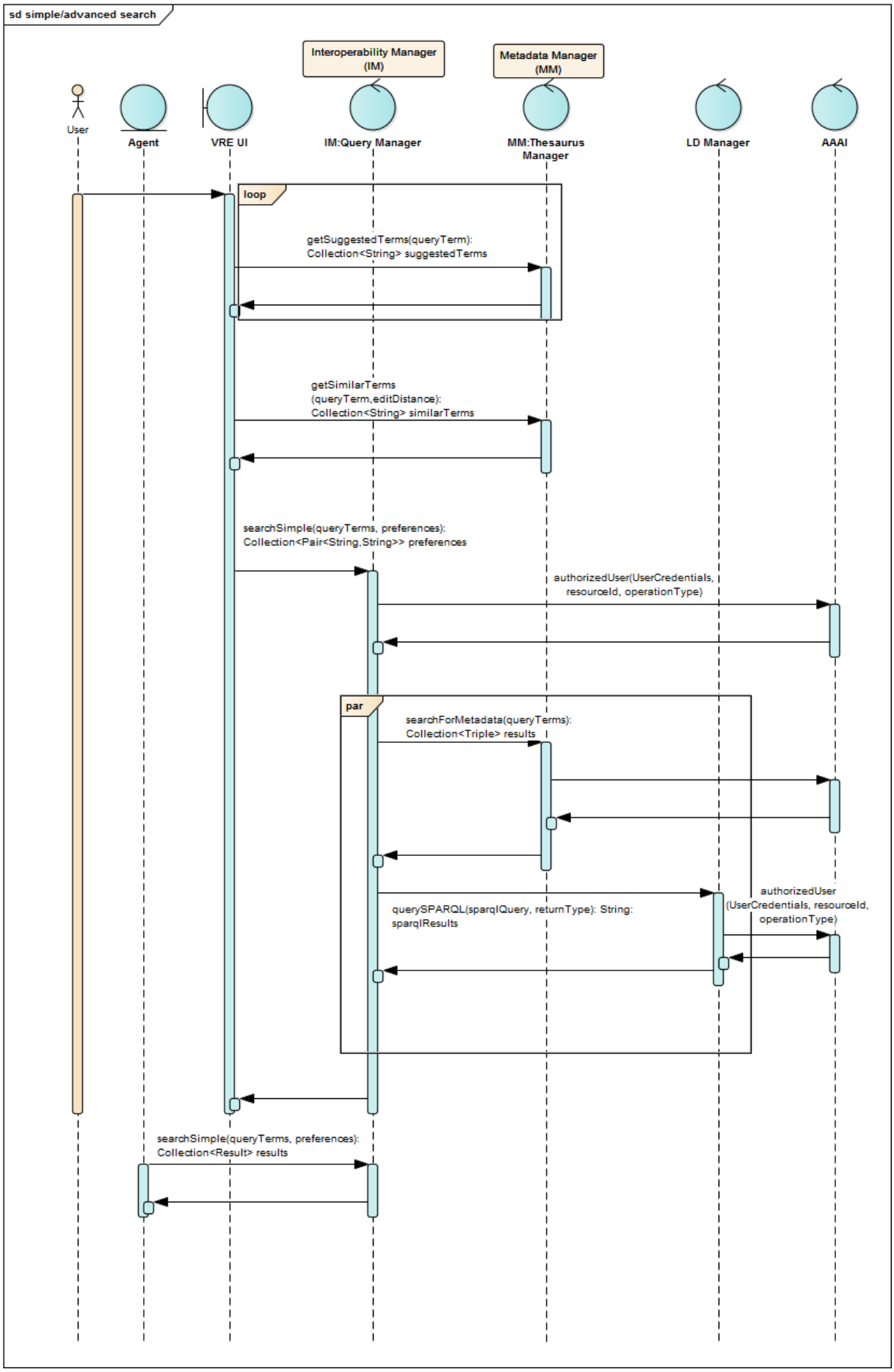
In particular, it should be clear enough to specify how the various user activities will be supported by the underlying architecture and the interplay of components, in order to assess the adequacy of the

designs. For that purpose, we hereby present four commonly met, domain-independent activities performed by VRE users, namely simple searching within the VRE, cross searching among various eRIs, browsing on the retrieved data and data publishing. We provide sequence diagrams that contain class, interactions, dependencies and activities among components, helping in making a clearer connection between the domain-independent use cases, the sequence of steps that are involved and the components of the architecture that contribute towards accomplishing these steps.

Tables 2, 2.1 and 2.2 in the Annex, assist even further in creating a much more detailed picture of this correlation, describing the Generalized and the elementary Functions that are triggered while performing each step, as well as the requirements that are satisfied. This way, a complete cycle is filled, starting from the use cases and leading to the requirements, as these have been identified by the users.

6.2.1 Search (simple/advanced)

In general the simple search process includes three main steps. The first one is the query preparation, the second one is the query submission and the third one is the returning of the results. The Query Manager(QM) component has the main responsibility for this process for both cases: an agent which calls a search api or a human user which interacts with the VRE search UI. At first, the user types a set of terms and at the same time the Query Manager communicates with the Thesaurus component in order to provide suggested terms based on the typed keywords and suggests similar terms by proceeding to spell checking. There are some advanced options for the user in order to perform an advanced search by applying an extra specific number of search criteria (i.e target data source or dataset). Afterwards the query is submitted to the QM which examines the sources, breaks the query into subqueries and submits the queries to the MetadataManager and the LD Manager. Finally, the QM retrieves all results. In the case of an agent the QueryManager is called in order to return the results through an api call. *In this sequence diagram it is assumed that the user/agent has been authenticated by AAAI. Also another assumption is that user preferences have been extracted from his profile.*


sd simple/advanced search
Interoperability Manager (IM)
Metadata Manager (MM)
User
Agent
VRE UI
IM:Query Manager
MM:Thesaurus Manager
LD Manager
AAAI
loop
getSuggestedTerms(queryTerm): Collection<String> suggestedTerms
getSimilarTerms (queryTerm,editDistance): Collection<String> similarTerms
searchSimple(queryTerms, preferences): Collection<Pair<String,String>> preferences
authorizedUser(UserCredentials, resourceId, operationType)
par
searchForMetadata(queryTerms): Collection<Triple> results
querySPARQL(sparqlQuery, returnType): String: sparqlResults
authorizedUser (UserCredentials, resourceId, operationType)
searchSimple(queryTerms, preferences): Collection<Result> results

6.2.2 Cross search

The **cross search** process provides the ability of searching in different data sources and datasets and finally returns a collection of results in a unified format that has been selected at the beginning of the query.

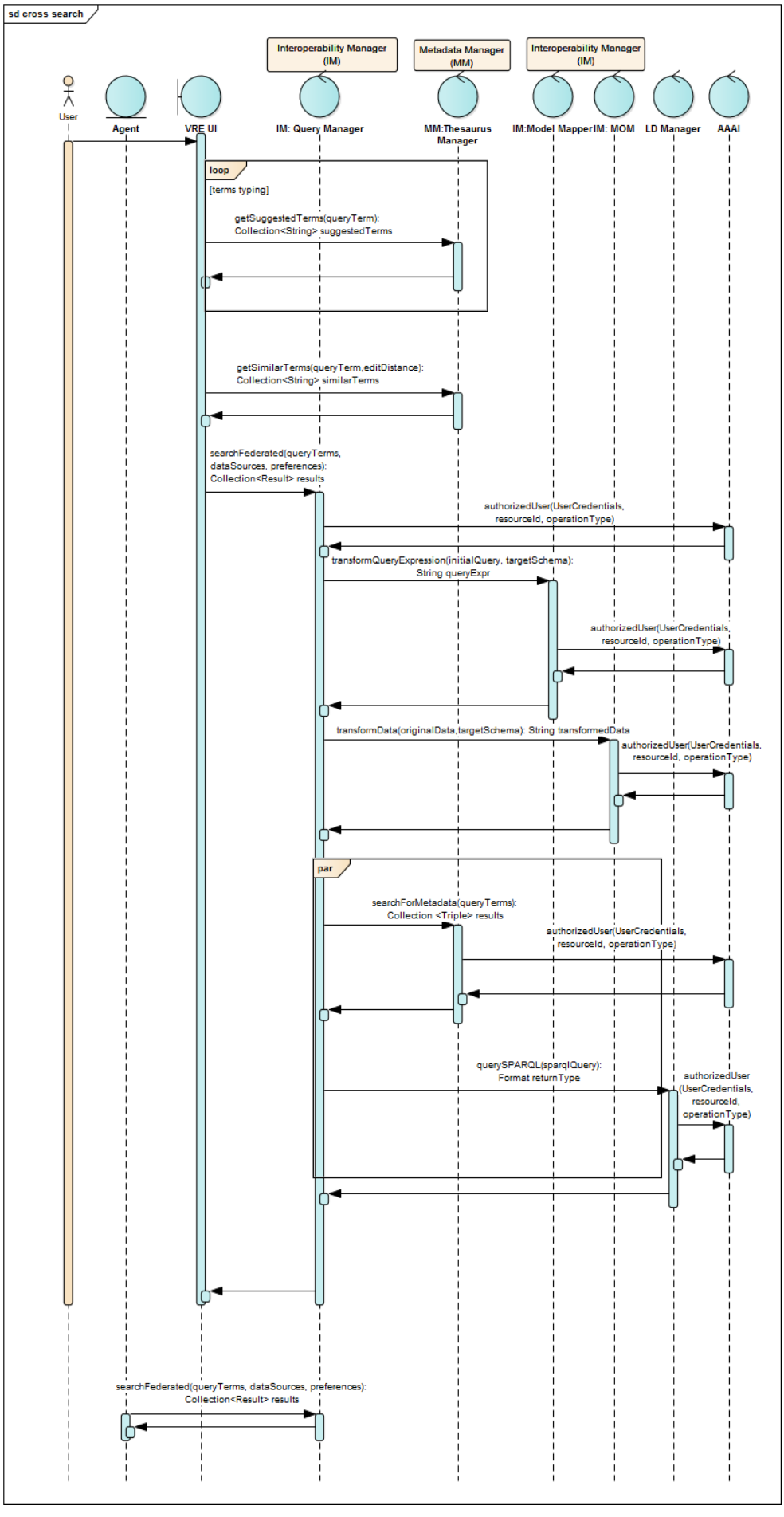
Similarly to the simple search for user case, the Query Manager communicates with the Thesaurus component for the query preparation.

The next step is the query transformation. Specifically, the QM communicates with the ModelMapper in order to perform a query translation to the desired model/schema format. Afterwards the appropriate subqueries which have been created in the previous step are sent to the corresponding external sources through the MOM component.

Subsequently, the federated query is submitted to the QM which also breaks the query into subqueries and submits the queries to the MetadataManager and the LD Manager in order to perform automatic corrections of keywords and recording the query evaluation provenance. Finally, the ModelMapper exploits the available mappings in order to perform the data source transformation and deliver the results in an homogeneous way to the users.

We should note that for performance reasons the transformation of data by the Model Mapper could be done per each source instead of doing it at the end. In the case of an agent case the QueryManager is called in order to return the homogeneous results through an API call.

In this sequence diagram, it is assumed that the requested access token has been granted for all selected sources by AAAI and the user explicitly has defined its selection to perform a cross query in the VRE

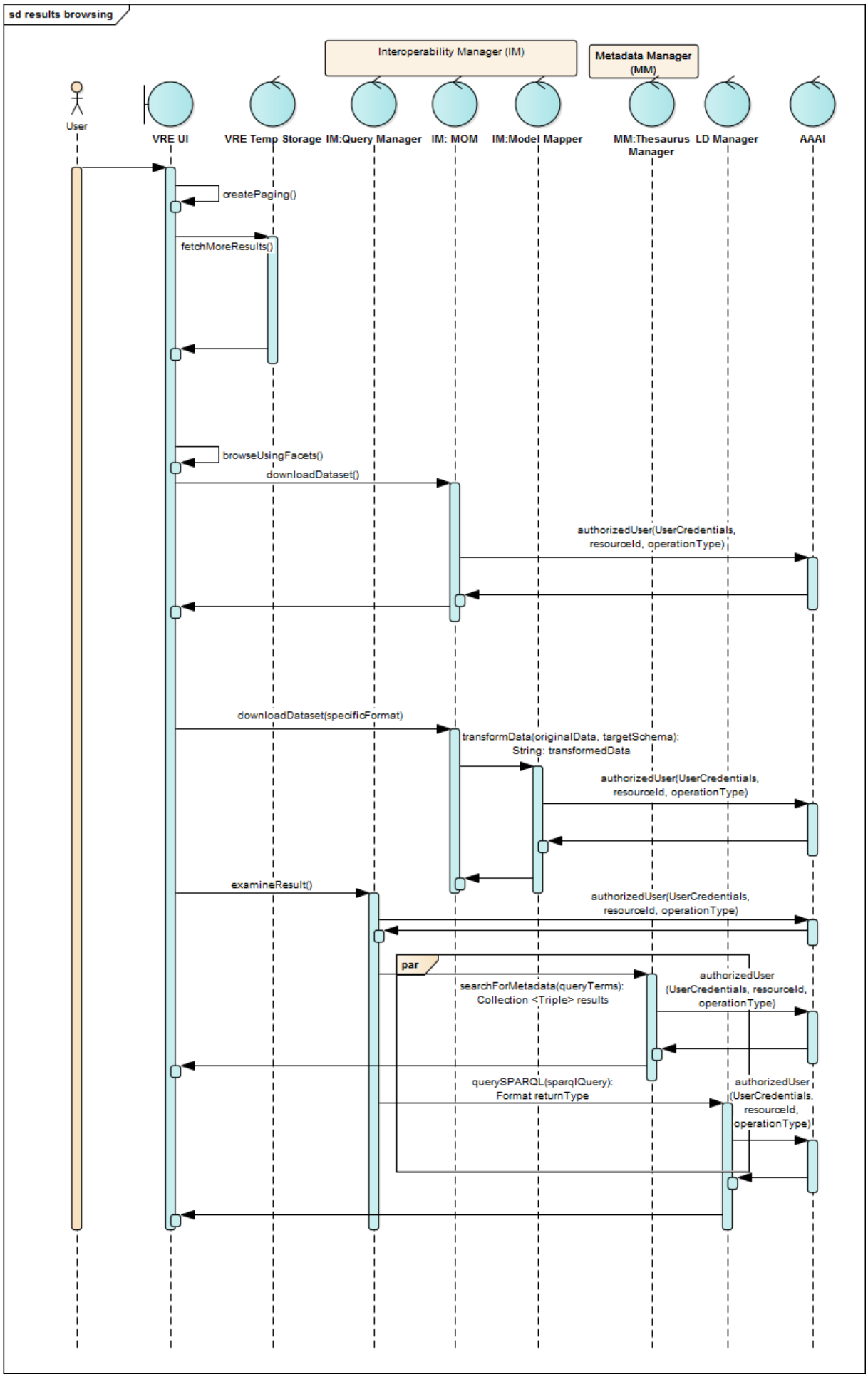

sd cross search
User
Agent
VRE UI
Interoperability Manager (IM)
IM: Query Manager
Metadata Manager (MM)
MM:Thesaurus Manager
Interoperability Manager (IM)
IM:Model Mapper
IM: MOM
LD Manager
AAAI
loop
[terms typing]
getSuggestedTerms(queryTerm): Collection<String> suggestedTerms
getSimilarTerms(queryTerm,editDistance): Collection<String> similarTerms
searchFederated(queryTerms, dataSources, preferences): Collection<Result> results
authorizedUser(UserCredentials, resourceId, operationType)
transformQueryExpression(initialQuery, targetSchema): String queryExpr
authorizedUser(UserCredentials, resourceId, operationType)
transformData(originalData,targetSchema): String transformedData
authorizedUser(UserCredentials, resourceId, operationType)
par
searchForMetadata(queryTerms): Collection <Triple> results
authorizedUser(UserCredentials, resourceId, operationType)
querySPARQL(sparqlQuery): Format returnType
authorizedUser (UserCredentials, resourceId, operationType)
searchFederated(queryTerms, dataSources, preferences): Collection<Result> results

6.2.3 Results browsing

The results browsing process relies on the interaction of the user with the corresponding components of the user interface, as well as with the components it is connected to. After performing a search using the QueryManager component, the user inspects the results. More specifically the user has the following capabilities:

- Browse over the results. The returned results are divided into pages so that the user can browse over them easily. So the user can start browsing over the results by selecting the available pages. All this interaction is being carried out from the UI components.
- Fetch more results. As soon as the user consumes all the available pages containing results (the top-K results), he/she can ask for more results from the VRE Temporary Storage component; this component is being used for storing all the results that are returned from the QueryManager component, however the UI component shows only the top-K results (for efficiency reasons). If there are more results to show then the UI is being updated and the new results are integrated with the initial ones. If there are no more results to show, then the user is being notified and the corresponding option (Fetch more results) is being disabled in the UI.
- Browse over the results using facets. The user can start browsing over the results by exploiting a set of facets (i.e. dataset type, latest modification date, etc.). These facets allow the user to restrict the results to those having the particular values. All the corresponding interactions are carried out by the UI components.
- Download a dataset. The user can select to download a dataset. For this reason the MOM component is exploited for retrieving and downloading the dataset.
- Download a dataset in a specific format. The user can select to download a dataset in a specific format. For this reason the MOM component is exploited for retrieving and downloading the dataset and the ModelMapper component (and in particular the Data Transformer subcomponent) for taking care of the transformation.
- Check for more information about a result. The user examines one results and wants to find more information about the results (more metadata). For this reason it communicates with the QueryManager component and search for more relevant information or related resources.

It is assumed that in every step the AAAI component has authenticated the user/agent and has authorized and given access according to the related permissions.


sd results browsing
Interoperability Manager (IM)
Metadata Manager (MM)
User
VRE UI
VRE Temp Storage
IM:Query Manager
IM: MOM
IM:Model Mapper
MM:Thesaurus Manager
LD Manager
AAAI
createPaging()
fetchMoreResults()
browseUsingFacets()
downloadDataset()
authorizedUser(UserCredentials, resourceId, operationType)
downloadDataset(specificFormat)
transformData(originalData, targetSchema): String: transformedData
authorizedUser(UserCredentials, resourceId, operationType)
examineResult()
authorizedUser(UserCredentials, resourceId, operationType)
par
searchForMetadata(queryTerms): Collection <Triple> results
authorizedUser (UserCredentials, resourceId, operationType)
querySPARQL(sparqlQuery): Format returnType
authorizedUser (UserCredentials, resourceId, operationType)

6.2.4 Data publishing

The data publishing process refers to the ingestion of the related metadata of data resources to the infrastructure's repositories to enable their *discovery* and *access*.

The first step of the data publishing process is the *acquisition* of the metadata. The metadata in their initial format can be provided either by a user/agent *a) by importing the complete metadata record files along or b) by filing a form in the UI*, or by automatically (and periodically) harvesting the metadata by the underlying sources (eRIs).

The metadata records are imported to the VREs workspace, along with the metadata standard that has been used for their representation.

If the metadata standard is different from the centralized underlying schema of the infrastructure (CERIF) then the ModelMapper component is called.

If there is a *mapping* in the ModelMapper's repository between the centralized schema and the metadata records schema, the metadata records are transformed with respect to the centralized schema.

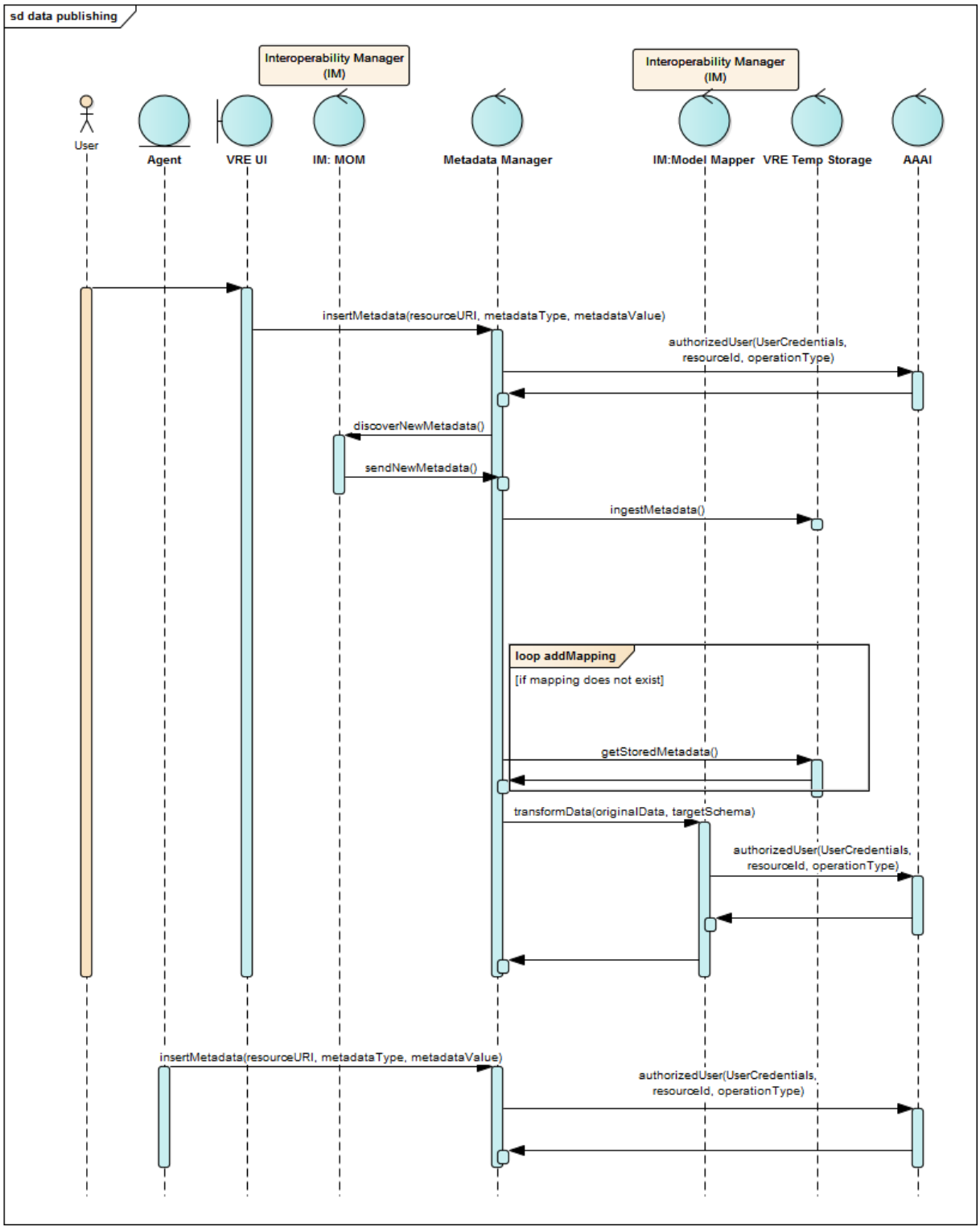
If the mapping does not exist then the user/agent creates the mapping between the two schemata, and the transformation of the records is applied according to the new mapping.

The final step of the data publishing process is the *ingestion* of the (transformed) metadata records to the metadata catalogue by the MetadataManager.

An intermediate storage layer for temporarily storing the uploaded metadata before transforming them into a proper format, is being exploited.

This layer is being provided through the VRE Temporary Storage component.

It is assumed that in every step the AAAI component has authenticated the user/agent and has authorized and given access according to the related permissions.


sd data publishing
Interoperability Manager (IM)
Interoperability Manager (IM)
User
Agent
VRE UI
IM: MOM
Metadata Manager
IM:Model Mapper
VRE Temp Storage
AAAI
insertMetadata(resourceURI, metadataType, metadataValue)
authorizedUser(UserCredentials, resourceId, operationType)
discoverNewMetadata()
sendNewMetadata()
ingestMetadata()
loop addMapping
[if mapping does not exist]
getStoredMetadata()
transformData(originalData, targetSchema)
authorizedUser(UserCredentials, resourceId, operationType)
insertMetadata(resourceURI, metadataType, metadataValue)
authorizedUser(UserCredentials, resourceId, operationType)

7 Outlook

The Reference Architecture presented in this deliverable represents one outcome of the first year of work of the project VRE4EIC. A final Reference Architecture will be produced at the end of the project, in two years from the submission of the present deliverable. This Section briefly describes the steps that will lead to the final architecture, highlighting the contributions from the various task that will be involved in the process during the remaining two years time.

A revised version of the Reference Architecture will be produced by the end of the second year of the project, for internal usage only. This revision has been deemed as necessary by the project management as a consequence of the fact that the requirements deliverable (D2.1) will be in turn refined three times during the second year of the project. These refinements have been planned in order to accommodate the collection of a large amount of requirements, along with the on-going characterization of existing e-RIs. Each time the requirements and the e-RI characterizations will be updated, the Reference Architecture will consequently be revised. The revision task will be greatly simplified by the fact that the architecture development team has maintained the traceability of the architecture, which links the requirements to the components in the architecture. Thus it will be possible to know which requirement is implemented by which interface, making it easier to take into account updated requirements.

In parallel to the revision of the architecture, an implementation phase will be carried out, comprised of three tasks:

- Task 3.2 – Gap Analysis Existing, until M21, to determine the components to be developed
- Task 3.3 – Development of Building Blocks, until M36, to develop the previously selected components
- Task 3.4 – Integration of Reference VRE and Enhanced Existing VREs, until M36, to integrated the implemented components into the architectures of EPOS and ENVRIPlus.

It is expected that both Task 3.3 and 3.4 will bring new insights that may lead at a revision of the Reference Architecture. In particular,

- in developing building blocks, the VRE4EIC development team will rely on the re-use of existing technologies and standards, which in turn may lead to the revision of some interfaces of the Reference Architecture, for instance to align an interface with the selected standard or technology; this alignment may be propagated into the Reference Architecture, if the standard or technology provoking it are important enough;
- In integrating the Reference Architecture with EPOS or ENVRIPlus, the VRE4EIC development team may need to adapt some interface to the target architecture. As for the previous point, this adaptation may be propagated into the Reference Architecture, if it enhances the quality of the Reference Architecture with respect to the principles outlined in Section 2.

Finally, a revision of the Reference Architecture may be fired by the developments in WP4 on Interoperability or WP5 on Trust, Security and Privacy. In particular,

- Work in WP4 may lead to a revision of CERIF, the data model which the Reference Architecture is based. Consequently, some interfaces in the Reference Architecture may have to be modified. It is worth mentioning that also the reverse may happen: the development of the Reference Architecture may bring up some representational inadequacies of CERIF and lead to an enhancement of the model to cope with those inadequacies. This has not happened during the development of the present Reference Architecture, but may happen in any one of its refinements, described above.
- Work in WP5 may lead to a revision of the AAAI component of the Reference Architecture.

8 Conclusions

The VRE4EIC initial Reference Architecture has been provided by the present deliverable, and contextualized in several ways.

The methodology followed in deriving the architecture has been first illustrated, to the end of connecting the work reported here with the literature and the approaches for architecture development in the context of software engineering.

The analysis of requirements leading to the architecture has been subsequently reported, to the end of grounding the architecture to the needs that it is expected to respond to. The traceability of the architecture has also been derived, linking the architecture's components to the requirements. These links will be necessary to manage in an optimal way the evolutions of the Reference Architecture leading to the final architecture, to be delivered at the end of the project.

The Reference Architecture has been provided, in terms of three kinds of UML diagrams:

- A main component diagram highlighting the components of the architecture, their provided and used interfaces;
- A series of sequence diagrams highlighting the interactions occurring in the execution of the main methods.

A main usage scenario of the architecture has been illustrated by showing the steps need for e-RIs to join an existing eVRE.

Finally, the roadmap from the current architecture to the final architecture has been outlined, showing the contributions expected from the various project activities that may provide useful inputs or elements to the architecture.

9 Annexes

9.1 Architectural components

Table 1. e-VRE conceptual components and sub-components

Component ID	Subcomponents/ Datasets	Description
Authentication, Authorization, Accounting Infrastructure (AAAI)		Authentication, Authorization, Accounting Infrastructure
Metadata Manager (MM)		Manages metadata about e-VRE entities: resource descriptions, user descriptions, provenance information, preservation metadata etc. (CERIF)
	User Catalogue	Contains user profiles and preferences
	Resource Catalogue	Contains metadata about resources available in e-VRE, i.e. datasets, services, workflows, instruments, networks of sensors, software applications etc
	Preservation Catalogue	Contains information related to the preservation process
	Provenance Catalogue	Contains metadata related to provenance
Interoperability Manager (IM)		Manages interactions with e-RIs
	Query Manager (QM)	Manages local and distributed queries, collects result sets
	Data Model Mapper (DMM)	Manages data and query format conversion
	Adapters	Ad hoc components, that synchronously interacts with e-RIs resources
	Message-Oriented Middleware (MOM)	Manages asynchronous interactions with eRIs resources using messaging protocols
	e-VRE Web Services (e-VRE WS)	Published to enable external applications to interact with e-VRE

Workflow Manager (WM)		Manages business processes and and scientific workflows, information about workflows are stored in the Metadata Manager
	Workflow configurator	Provides functionalities: to build/edit/store *execution plans,* to control and monitor processing flows execution.
	Workflow executor	Manages workflow execution, including data staging
	Workflow repository	Provide functionalities to store and retrieve workflows, workflows will be published using LD manager
Linked Data Manager (LDM)		Manages the publication of information in e-VRE as Linked Data
	SPARQL Endpoint	The SPARQL service endpoint allows retrieving resources and services published by e-VRE as RDF documents
	LD API	The LD API: maps CERIF metadata records in RDF, implements metadata enrichment of RDF records, i.e. adds to records typed links to vocabularies and thesaurus entries, Implements content negotiation
System Manager (SM)		Implements functionalities to *define* and manage the VRE, e.g. specify the resources, specify the apps, and to deploy the defined VRE in the available resources.
	Node Manager (NM)	Implements the functionalities to deploy, manage and run an instance of e-VRE on a specific hardware
	User Manager (UM)	Manages user profiles and provides collaboration/communication functionalities for users. It provides the functionalities to add/update/remove user profiles, to set up users permissions, to manage users preferences, to configure users working environments
	Resource manager (RM)	Manages resource information: add/update/remove resource descriptions, associate resources to security policies, etc.
	App Manager (AM)	This component provides functionalities to deploy and manage applications that operates on e-VRE resources. It can be used also to embed applications such as Wiki or forums etc.

9.2 Generalised functions

The tables in this subsection clarify the connection between the different viewpoints already described in the document. Specifically, as identified in the use cases, there are a number of domain-independent, general tasks that a user of a VRE typically performs, such as querying for metadata, asserting information in the catalogues, and others. These are called *Generalized Functions*. Such Generalized Functions are usually composed of elementary *Functions,* e.g., cross-searching, which are structured based on specific series of steps. The elementary Functions have a direct connection with the *Requirements*, as they have been developed, in order to satisfy one or more of them. Notice that even elementary Functions can be further decomposed into other functions, according to the level of abstraction needed.

Table 2 connects the Generalized Functions extracted from the use cases with the elementary Functions and the steps involved. Then, Tables 2.1 and 2.2 below elaborate on the elementary Functions, associating them with the relevant requirements of each. In addition, they correlate these functions with the corresponding components of the architecture which implements them, as already explained in sections 3 and 4.

Table 2. Generalised Functions from Domain-Independent Use Cases and Requirements

Generalized Functions	**Pre-condition**	**Steps Involved**	**Included/ Specialized by Fun**
GF1: Querying	**Fun21**: Agent Authentication (the agent has been succesfully authenticated)	● The agent accesses the VRE Search UI (in case of a human user) or the corresponding Search API (in case of software entity) ● The agent prepares a query to be submitted, involving ○ keywords, and/or ○ topic filtering, and/or ○ target (internal or external) sources, and/or ○ target datasets, and/or ○ other filtering criteria ● During the query preparation process, the system offers spell checking and recommendations (Thesaurus - MM) ● The query is submitted to the QueryManager by the agent. ● The Query Manager receives a query and starts parsing it. Specifically: ○ The Query Manager (QM) examines the sources ○ The QueryManager breaks the query into subqueries ○ The QueryManager submits the queries (to MetadataManager and LD Manager)	**Fun1**:Simple Search **Fun2**:Cross Search **Fun3**: Advanced Search (the next ones are neither primitive nor abstract; they are compound functions) **Fun11**: Data Collection **Fun12**: Data Sampling **Fun19**: Data Discovery

		● The QueryManager retrieves all results ○ The QM merges the results ○ The QM returns the integrated results ● The agent browses over the results (IDs of resources) (see GF2: Dataset/Metadata Exploration)	
GF2: Dataset/ Metadata Exploration	**GF1**: Querying	● The agent has a list with IDs of resources. ● The UI shows the results to the user (only for human users) ○ The UI creates the pages with the results ○ The UI shows some information about the query evaluation (might require communicating with other components for retrieving such information) ● The user selects to browse for more information about a specific result ● The above request is submitted to QueryManager, along with the selected source (from GF1) ● The QueryManager inspects the given sources and submits the corresponding requests to MetadataManager and LDManager ● The QueryManager returns the integrated results as a graph ● The UI shows the graph to the user (only for human users)	**Fun4**:Dataset Viewing **Fun5**: Dataset Preselection **Fun6**: Dataset Customization (the next ones are neither primitive nor abstract; they are compound functions) **Fun19**: Data Discovery **Fun11**: Data Collection **Fun12**: Data Sampling
GF3: eRISoftware	Access to a eRI service through VRE (e.g., instrument-related or other non-VRE service)	● The agent accesses the menu or UI or APIs that contains the list of instruments/real-time sensor data/processes/third-party software etc ● The agent selects the one it desires. ● The system redirects the agent to the eRI interface ● The agent interacts directly with a dedicated UI (e.g., it accesses/calibrates/configures instruments) ● The system maintains a timestamped log with the user's actions	**Fun7**: Instrument Integration **Fun8**: Instrument Configuration **Fun9**: Instrument Calibration **Fun10**: Instrument Monitoring (the next ones are neither primitive nor abstract; they are compound functions) **Fun11**: Data Collection **Fun12**: Data Sampling
GF4: Data Cataloguing		● The user wishes to register a new resource to the VRE catalogue or update an existing one ● She selects the type of	**Fun13**: Resource Registration **Fun14**: Resource Update

		resource, in order to be redirected to the appropriate UI ● She fills in forms asking for the necessary metadata. Some fields are already completed by the system. ● The system checks the metadata and returns the corresponding messages ● The user performs quality improvement ● The user selects to store the metadata to the catalogue ● (if needed) The system communicates with the underlying eRI to verify that the dataset involved have been stored successfully ● If the previous step is successful, the system stores the metadata to the corresponding catalogue ● The system also updates the preservation and provenance catalogues as appropriate	**Fun17**: Resources Annotation **Fun18**: Metadata Harvesting
GF5: Workflow Enactment		<TBD> ● The user builds the workflow using the UI of the Workflow Manager ● The workflow is deployed on the execution engine(s) ● The workflow is executed ● The WF Manager monitors the workflow tasks and notify the user for registered update ● The results are stored in a temporary area accessible by the user	**Fun15**: Workflow Enactment (the next ones are neither primitive nor abstract; they are compound functions) **Fun12**: Data Sampling

Table 2.1 Query Requirements

Function ID	**Requirement ID**	**Function Description**	**Involved components**	**Related Generic Functions & Notes**
Fun1: Simple Search	**PRQ10** Simple search **PRQ11** Multiple format support **PRQ14** Spelling checking	**The user performs a simple search** 1. The user inserts in a form a list of keywords 2. The user submits the form 3. The system retrieves the query results and returns them to the user	**UI** **IM** Query Mediator Query Manager **MM**	**GF1**: Querying

	PRQ15 Query suggestion **IRQ5** Citation Tracking		Resource Catalogue **LD** **AAAI**	
Fun2: Cross Search	**PRQ12** Cross searching **PRQ14** Spelling checking **PRQ15** Query suggestion **PRQ20** Linking external resources **IRQ5** Citation Tracking	**The user performs a cross search** 1. The user ticks on the cross searching option 1.1. (optional): The user defines a list of external sources 2. The user inserts in a form a list of keywords 3. The user submits the form 4. The system retrieves the query results and returns them to the user	**UI** **IM** Query Manager **MM** Resource Catalogue **LDM** **AAAI**	**GF1**: Querying
Fun3: Advanced Search	**PRQ13** Advanced search **PRQ14** Spelling checking **PRQ15** Query suggestion **PRQ16** Query Filter **IRQ5** Citation Tracking	**The user performs an advanced search** 1. The user clicks on the advanced search option 2. The user selects/edits a number of search criteria 3. The user inserts in a form a list of keywords 4. The user submits the form 5. The system retrieves the query results and returns them to the user	**UI** **IM** Query Manager **MM** Resource Catalogue Provenance Catalogue Preservation Catalogue **LDM** **AAAI**	**GF1**:Querying
Fun4: Dataset Viewing	**PRQ17** Datasets viewing	**The user views all the datasets metadata** 1. The user performs simple search without submitting any keywords 2. The system retrieves the full list of datasets' metadata	**UI** **IM** Query Mediator Query Manager **MM** Resource Catalogue **LDM** **AAAI**	**GF2**: Dataset/ Metadata Exploration

Fun5: Dataset Preselection	**PRQ18** Datasets pre-selection	**The user pre-selects datasets** 1. The user views all the datasets (PRQ17) 2. The user selects the datasets to search on	**UI** **IM** Query Mediator Query Manager **MM** Resource Catalogue **LDM** **AAAI**	**GF2**: Dataset/ Metadata Exploration
Fun6: Dataset Customization	**PRQ19** Dataset customization	**The user selects a set datasets to be exploited during search** 1. The user selects his profile 2. The user views all the datasets (PRQ17) 3. The user selects the datasets to search on	**UI** **IM** Query Mediator Query Manager **MM** Resource Catalogue User Catalogue **LDM** **AAAI**	**GF2**: Dataset/ Metadata Exploration

Table 2.2 Data Requirements

Function ID	Requirement ID	Function Description	Involved components	Related Generic Functions & Notes
Fun7: Instrument Integration	**CLRQ1** Instrument Integration	**The user views many instruments** 1. The user selects to views all the available instruments 2. The system returns an integrated list of instrument description to the user	**UI** **IM** Query Mediator Query Manager **MM** Resource Catalogue **LDM**	**GF3**: eRISoftware

			AAAI	
Fun8: Instrument Configuration	**CLRQ2** Instrument Configuration **CLRQ4** Instrument Access **CLRQ5** Configuration Logging **CLRQ10** Process Control	**The user configures an instrument** 1. The user selects an instrument 2. The system redirects the user to the RI interface 3. The user configures the instrument (The RI creates the appropriate logs)	**UI** **SM** Resource Manager **MM** Resource Catalogue **IM** **AAAI**	**GF3**: eRISoftware
Fun9: Instrument Calibration	**CLRQ3** Instrument Calibration **CLRQ4** Instrument Access	**The user calibrates an instrument** 1. The user selects an instrument 2. The system redirects the user to the RI interface 3. The user calibrates the instrument	**UI** **SM** Resource Manager **MM** Resource Catalogue **IM** **AAAI**	**GF3**: eRISoftware
Fun10: Instrument Monitoring	**CLRQ6** Instrument Monitoring **CLRQ7** (Parameter) Visualisation **CLRQ8** (Real-Time) (Parameter/Data) Visualisation	**The user monitors an instrument** 1. The user selects an instrument 2. The system redirects the user to the RI interface 3. The RI returns information to the user on the instrument's status/parameters/data	**UI** **SM** Resource Manager **MM** Resource Catalogue **IM** **AAAI**	**GF3**: eRISoftware
Fun11: Data Collection	**CLRQ11** Data Collection **CLRQ12** (Real-Time) Data Collection	**The user retrieves data from an instrument** 1. The user performs a cross searching advanced search 2. The system returns a list of results and the hosting RIs 3. The user retrieves/collects the data from the RIs	**UI** **IM** Query Manager **MM** Resource Catalogue User Catalogue	**GF1**: Querying **GF2**: Dataset/ Metadata Exploration **GF3**: eRISoftware

			Resource Catalogue **SM** Resource Manager **LD<** **AAAI**	
Fun12: Data Sampling	**CLRQ13** Data Sampling	**The user performs data analysis** 1. The user performs a cross searching advanced search 2. The system returns a list of results and the hosting RIs 3. The user retrieves/collects the data from the RIs 4. The user selects the analysis to be performed (workflow) 5. The system returns the analysis results	**UI** **SM** Resource Manager **MM** Resource Catalogue User Catalogue **IM** Query Manager **WM** **AAAI**	**GF1**: Querying **GF2**: Dataset/ Metadata Exploration **GF3**: eRISoftware **GF5**: Workflow Enactment
	CTRQ9 Online Dataset Editing **CLRQ14** Noise Reduction **CLRQ15** Data Transmission **CLRQ16** Data Transmission Monitoring **PVRQ1** Data Provenance **PVRQ2** Data Acquisition Information	**RIs responsibility**		

	PVRQ3 Data Curation Information **PVRQ4** Data Publication Information **IRQ1** Data Identification **IRQ3** Raw Data Identification **IRQ4** Data Citation **CRQ1** Data Product Generation **CRQ7** Data Replication **CRQ8** Replica Synchronisation **PRQ7** Data Compression **PRQ29** Data Processing Monitoring **PRQ35** Data Backup			
Fun13: Resource Registration	**CLRQ17** Data Cataloguing **CRQ2** Data Quality Checking **CRQ3** Data Quality Verification **CRQ6** Data Storage &	**The user registers a resource performing (meta) data quality checking** 1. The user selects to register/update a new resource in the catalogue 2. The system checks the metadata and returns the corresponding messages 3. The user performs the quality improvement 4. The user selects to store the metadata to the catalogue 5. The system stores the metadata to the corresponding	**UI** **SM** Resource Manager **MM** Whole **IM** DMM **LDM**	**GF4**: Data Cataloguing

	Preservation **PRQ4** Resource Registration **PRQ5** (Metadata) Registration **PRQ6** Data Conversion **CLRQ9** Experiment **CTRQ15** Funding body Information **CLRQ18** Data Publication **IRQ2** Data Provider **IRQ4** Data Citation **PRQ8** Semantic Harmonisation	catalogues	**AAAI**	
Fun14: Resource Update	**CRQ4** Data Versioning	**The user registers a new version of a resource** 1. The user selects to register a new resource version in the catalogue 2. The system updates the resource, preservation and provenance catalogues accordingly	**UI** **SM** Resource Manager **MM** Whole **IM** DMM **LDM** **AAAI**	**GF4**: Data Cataloguing
Fun15: Workflow Enactment	**CRQ5** Workflow Enactment **PRQ26** Scientific	**The user creates a workflow** 1. The user creates a workflow 2. The user select to run the workflow 3. The system returns the workflow's results	**UI** **WM** **MM**	**GF5**: Workflow Enactment

	Workflow Enactment **PRQ28** Data Processing Control **ORQ2** Processing Parallelisation **ORQ4** Data Compartmentization		**IM** **AAAI**	
Fun16: Access Control	**CRQ6** Data Storage & Preservation	**Fundamental Infrastructure's Functionality**	**MM** **AAAI**	
Fun17: Resources Annotation	**PRQ1** Resources Annotation **PRQ2** (Data) Annotation **PRQ32** Quality Rating **PRQ34** Data Tag	**The user annotates data/resource** 1. The user selects a resource 2. The user annotates the resource 3. The system stores the appropriate information	**UI** **SM** Resource Manager **MM** Resource Catalogue Metadata Catalogue	**GF4**: Data Cataloguing
Fun18: Metadata Harvesting	**PRQ3** Metadata Harvesting	**The user enables metadata harvesting** 1. The user selects a source to be harvested 2. The user selects the frequency 3. The system harvests the metadata and either registers a new resource or updates an existing one's metadata	**UI** **SM** Resource Manager **MM** Whole **IM** DMM **LDM** **WM**	**GF4**: Data Cataloguing
Fun19: Data Discovery	**PRQ9** Data Discovery and Access **PRQ31**	**The user discovers data using the searching capabilities (see Query Requirements)**	**UI** **IM** Query Manager	**GF1**: Querying **GF2**: Dataset/ Metadata Exploration

	Dataset Download		**MM** Resource Catalogue **LDM**	
Fun20: User/ Agent Registration	**CTRQ4** Interface Customization **CTRQ5** Wizard Configuration **CTRQ7** Multilingual Interface	1. On the user device: User goes to eVRE portal in the browser, and creates her/his own user profile using functionalities provided by User Manager. 2. The User Manager interacts with MM to register the profile and gets from the MM the configuration information needed to configure the user environment 3. Once the profile is created UM asks "Do you want to register this device with eVRE as authenticator?" User agrees 4. Depending on the device UM can ask the user to download and install a software implementing VRE4EIC authentication API (to enable the device to implement authenticator client functionalities) 5. UM registers the device in MM and AAAI as authenticator 6. UM shows message, "Registration complete."	**UM** **AAAI**	For details on the authentication and registration framework adopted in eVRE see the section Notes on the E-VRE authentication protocol above
Fun21: Agent Authentication	**CTRQ1** Login **CTRQ31** Accounting	**1) User Authentication (external authenticator mode):** - User access E-VRE in browser, sees an option "*Sign in with your registered device.*" - User chooses this option and gets a message from the browser, "*Please complete this action on your phone.*" -**On the phone**: - User sees a notification similar to "Sign in to eVRE?" - User selects this notification. - User is shown a list of their VRE4EIC identities, e.g., "*Sign in as Alice / Sign in as Bob.*" - User picks an identity and provides it. -**On the laptop**:	**UI** **UM** **AAAI** **MM (Fun)**	

		- Web browser shows that the selected user is signed in, and navigates to the signed-in page/restore the user session etc. **2) Device authentication (<u>external</u> authenticator mode)** - User connects an external instrument to a device or a network connected to eVRE - The device sends a message to eVRE to notify that he wants to add a new device to the Research Environment - On the user laptop/smartphone signed to eVRE,user sees a prompt similar to "*Authorise instrument to sign in on eVRE, chose identity*" - User is shown a list of their eVRE identities, e.g., "*Sign in as Alice / Sign in as Bob"*. - User picks an identity and provides it. - Instrument is ready to operate on eVRE.		
Fun22: Continuous Access	**CTRQ2** Continuous Access	1)**Continuous access: save session Use Case (session explicitly closed)** - Agent authenticates on eVRE - if present, the previous session is restored - agent operate on eVRE - Agent invoke 'sign out' functionality in the UM GUI, or specific 'save session' functionality - UM saves all information related to the current session in a specific storage assigning an identifier to the saved session - UM interacts with the MM manager in order to: i) associate the session identifier to the User Profile and ii) change the status of the User profile in case of sign out - If required UM interacts with AAAI, this could be required when	**UI, UM, AAAI, MM**	This requirement has been interpreted as: 'maintain user's session across multiple connections'

		we need to notify to external eRI that the connection is closed and the user is no longer operating on their resources **2) Continuous access: save session Use Case (session not explicitly closed)** **-** Agent authenticates on eVRE - if present, the previous session is restored - Agent operate on eVRE - For a defined amount of time no action occurs - UM automatically saves all information related to the current session in a specific storage assigning an identifier to the saved session - UM interacts with the MM manager in order to associate the session identifier to the User Profile **3) Continuous access: restore session Use Case** - Agent authenticates on eVRE - The User Manager interacts with Metadata Manager to discover if a session identifier is associated to the user profile - If an id is returned the information is retrieved from the specific storage - The UM interacts with the AAAI to verify that the user can access the content of the old session-is restored -Agent operates on eVRE		
Fun23: Update Alert	**CTRQ8** Update Alert **CTRQ10** Notification	**1) Update Alert: event subscription** - User authenticates on eVRE - The user uses User Manager to subscribe to events - The UM saves subscriptions on the User Profile (MM) and associate the id of the user to the list of users subscribed to that event in the event list **2) Update Alert: update notification**	**UI, MM, UM, RM, IM, AAAI**	An user receives a notification when an event she is interested in occurs. Generally speaking there could be a large number of events in a eVRE, to name a few: changes in the lifecycle of

		- User authenticates on eVRE - The user session is restored - The user uses the UM functionalities to read the alerts related to her/his subscriptions.		processes, changes in the availability status of a resource published by a RI, updates of a shared documents etc.
Fun24: Resource Connection	**CTRQ28** Computing Resource Connection	**Access EGI computational resources via OCCI** The user has previously obtained a VOMS certificate (http://www.eu-emi.eu/training/cert-tutorial) via AAAI functionalities, this information is registered in the user profile and the actual certificate is managed by AAAI. - User authenticates on eVRE - The user session is restored - User selects the EGI entry point - The user sends a request, validated with the VOMS certificate, to the EGI entry point asking for computational resources - The user chose the references to the computational resources and use them	**UI, IM, AAAI, MM**	This requirement means the possibility to access computational or storage resources for an application installed in the eVRE. These resources can be provided by eVRE environment or by an external provider. This requires that the user has permission to use those resources and there is an Adapter in the IM that interacts with those resources,

Notes on the E-VRE authentication protocol

This section contains a short description of the authentication protocol adopted in VRE4EIC, in order to explain the infrastructure underlying the Functions 20 and 21 described in the table above.
The E-VRE authentication protocol will be based on scoped credentials assigned to a User or an Agent and controlled by *authenticators* (https://en.wikipedia.org/wiki/Authenticator).
Please note that we are referring here to the authentication process between a user and the E-VRE system, it will 'wrap' the protocols adopted by VRE4EIC AAAI infrastructure.

The scoping of the credentials will be enforced jointly by a User Agent implementing the E-VRE authentication API and an authenticator that holds the credential, by constraining the availability and usage of credentials.

Scoped credentials are located on authenticators, which can use them to perform operations subject to user consent.
According to outcome of D2.1 we should have two types of authenticator:

- Authenticators located in the same device (e.g., smartphone, tablet, desktop PC) as the user agent is running on.
- Authenticators operate autonomously from the device running the user agent, and accessed via network or other protocols. This last part is needed mainly to implement requirements about external instruments or devices.

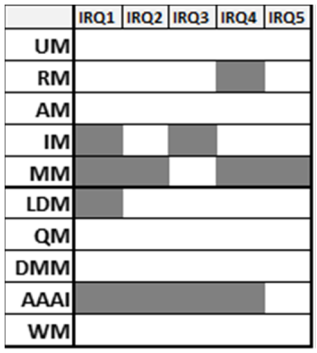
More than one user profile can be managed by a single authenticator: we need functionalities to add user profiles to an authenticator.

9.3 Requirements and components

The tables in this Section show the components involved in the implementation of user requirements. They have all the same structure, and each of them reflect a pillar of requirements. For every requirement pillar, the table highlights: the relationships with other requirements, the relationships with the metadata catalogue ("CERIF entities" column) and notes and comments that we have on the requirement.

Before each table, a diagram is given reporting the incidence matrix between the components in the Reference Architecture (row) and the requirement (column).

Data Identification and Citation



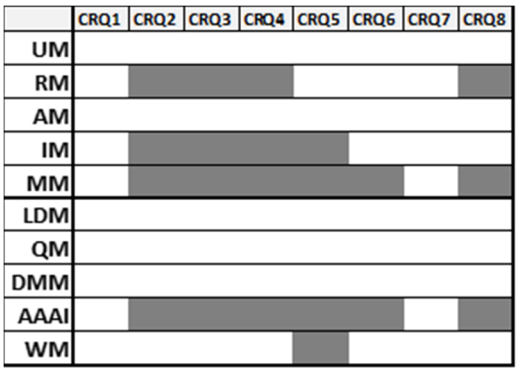
UM - User Manager, RM - Resource Manager, AM - App Manager, IM - Interoperability Manager, MM - Metadata Manager, LDM - Linked Data Manager, QM - Query

Table 3.1 Data Identification and Citation

Req. ID	Description	Rel. with other req.	Components involved	CERIF entities	Notes and Comments
IRQ1	Data	CLRQ17	AAAI, MM, IM,	cfResultPr	

	Identification		LDM	oduct (dataset), associated cfFedId	
IRQ2	Data Provider		AAAI, MM	cfOrgUnit_ ResultPrid uct	
IRQ3	Raw Data Identification	IRQ1	AAAI, IM		
IRQ4	Data Citation	IRQ1	AAAI, RM, MM		
IRQ5	Citation Tracking		MM		Through the exploitation of the Provenance Catalogue

Data Curation



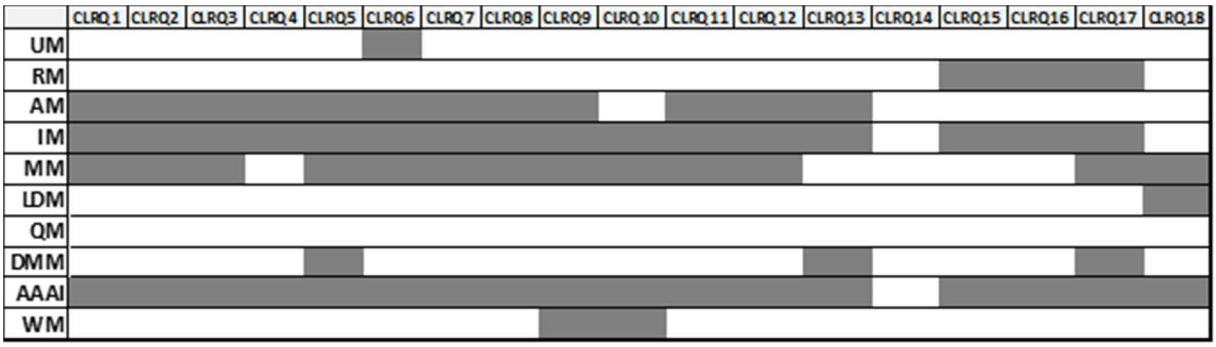
UM - User Manager, RM - Resource Manager, AM - App Manager, IM - Interoperability Manager, MM - Metadata Manager, LDM - Linked Data Manager, QM - Query

Table 3.2 Data Curation

Req. ID	Description	Rel. with other req.	Components involved	CERIF entities	Notes and Comments
CRQ1	Data Product Generation				eRI's responsibility
CRQ2	Data Quality Checking		RM, AAAI, MM, IM		Some checking and verification is needed before including data descriptions in our catalogues
CRQ3	Data Quality Verification	CRQ2	AAAI, RM, MM, IM		

CRQ4	Data Versioning		AAAI,RM, MM, IM, LDM		Only metadata update and versioning is our responsibility
CRQ5	Workflow Enactment		AAAI, WM, MM, IM		
CRQ6	Data Storage & Preservation		AAAI, MM		we only deal with the metadata here
CRQ7	Data Replication				eRIs responsibility
CRQ8	Replica Synchronisation		AAAI, RM, MM		eRIs responsibility

Data Cataloguing



UM - User Manager, RM - Resource Manager, AM - App Manager, IM - Interoperability Manager, MM - Metadata Manager, LDM - Linked Data Manager, QM - Query

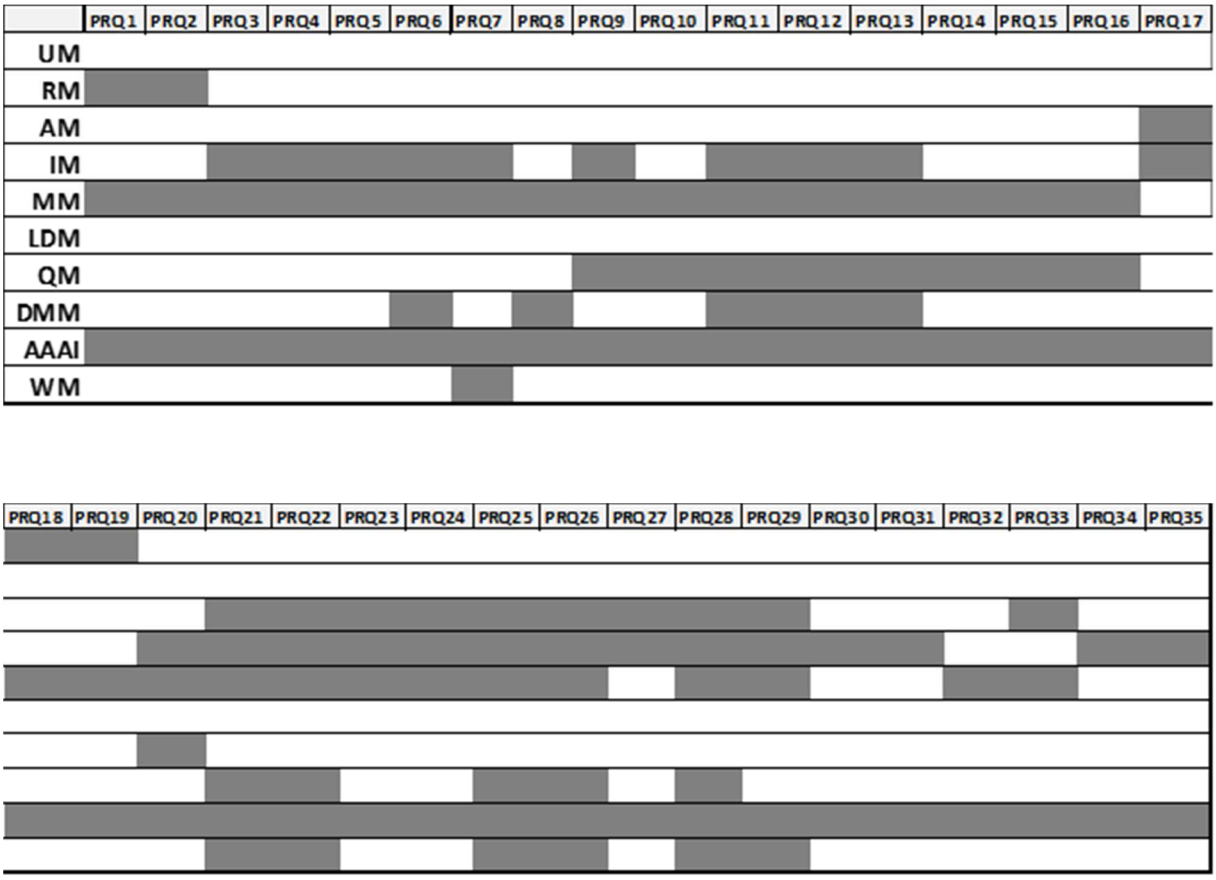
Table 3.3 Data Cataloguing

Req. ID	Description	Rel. with other req.	Components involved	CERIF entities	Notes and Comments
CLRQ1	Instrument Integration		AAAI,AM, MM, IM, LDM		CERIF has an equipment and a measurement entity For this set of requirements (CLRQ) we have to match them to the ENVRI 6 pillars. we assume we access the eRI functionalities, we only say to the users what they

					can do and instruct the eRIs to start their sensors etc. We act as mediators
CLRQ2	Instrument Configuration		AAAI,AM,MM, IM		
CLRQ3	Instrument Calibration	CLRQ2	AAAI,AM, MM, IM		linked with CLRQ2
CLRQ4	Instrument Access		AAAI,AM, IM		
CLRQ5	Configuration Logging		AAAI,AM, DMM, MM, IM		
CLRQ6	Instrument Monitoring	CLRQ4	UI,AAAI,AM, MM, IM		Uses CLRQ4
CLRQ7	(Parameter) Visualisation	Similar to CLRQ6	AAAI,AM, MM, IM		
CLRQ8	(Real-Time) (Parameter/ Data) Visualisation		AAAI,AM, MM, IM		May require data streaming analysis
CLRQ9	Experiment		AAAI, AM, MM, IM, WM		
CLRQ10	Process Control		AAAI, WM, MM, IM		
CLRQ11	Data Collection		AAAI, AM, MM, IM		A meta requirement. It contains a lot of features. It should be split into sub-requirements. Will affect provenance and preservation catalogues.
CLRQ12	(Real-Time) Data Collection	CLRQ11	AAAI, AM, MM, IM		Specialization of CLRQ11
CLRQ13	Data Sampling		AAAI, AM, DMM, IM		Can be modeled as real time integration with a statistical program. Other options could be: i) enable researchers to deploy and run their statistical programs on e-VRE, ii) e-VRE provides a set of statistical libraries Need to be defined.

					e-RIs responsible
CLRQ14	Noise Reduction				e-RIs responsible
CLRQ15	Data Transmission		AAAI, IM, RM		e-RIs responsible
CLRQ16	Data Transmission Monitoring		AAAI, IM, RM		e-RIs responsible
CLRQ17	Data Cataloguing		RI, AAAI, DMM, MM, IM		
CLRQ18	Data Publication		LDM, AAAI, MM		This requirement seems more generic

Data Processing



UM - User Manager, RM - Resource Manager, AM - App Manager, IM - Interoperability Manager, MM - Metadata Manager, LDM - Linked Data Manager, QM - Query

Table 3.4 Data Processing

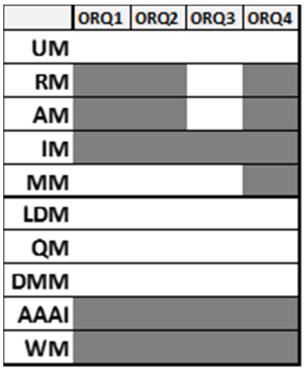
Req. ID	Description	Rel. with other req.	Components involved	CERIF entities	Notes and Comments
PRQ1	Resources Annotation		AAAI, RM, MM		
PRQ2	(Data) Annotation		AAAI, RM, MM		
PRQ3	Metadata Harvesting		AAAI, MM, IM		
PRQ4	Resource Registration		AAAI, MM, IM		
PRQ5	(Metadata) Registration	PRQ4	AAAI, MM, IM		
PRQ6	Data Conversion		AAAI, MM, IM LDM		Affects provenance and preservation catalogues
PRQ7	Data Compression		AAAI, MM, IM, WM		
PRQ8	Semantic Harmonisation		AAAI, DMM, MM		Consider http://www.w3.org/2005/Incubator/ssn/ssnx/ssn which is being formalised at http://w3c.github.io/sdw/ssn/
PRQ9	Data Discovery and Access	PRQ10, PRQ12, PRQ13	AAAI, MM, IM (QM)		Access is not our responsibility. Discovery is accomplished through metadata and the catalogue
PRQ10	Simple search	Uses: PRQ12	QM, AAAI , MM	Search into all / given attributes of CERIF metadata	Use opensearch.org for the query mediator. It is an interface. We should also guide users how to write queries (e.g., suggest queries) Simple search typically addresses the catalogue, other queries focus on the data themselves. The e-RI will also have their own search engines. How this integration will take place. Merging the results is not easy! The user may be asked how to complete the merging process (steer the process)

PRQ11	Multiple format support	Uses: PRQ12	QM, AAAI, MM, IM(DMM)	Metadata for cfResPubl (doc), cfMedium,(any other media), cfResProd (datasets)	
PRQ12	Cross searching		AAAI, MM, IM (DMM, QM)	Based on mappings CERIF - X	The query manager must know the query language of datasets involved and the mapper needs to know the result data formats, these info are stored on the Resource Manager
PRQ13	Advanced search	Uses: PRQ12	AAAI, MM, IM (DMM, QM)	Search into all / given attributes of CERIF metadata	There are 3 levels of queries, existential (generic), contextual (catalogue) and queries on the data. The latter is very difficult, we may decide that we only give connections to the underlying search facilities of eRIs.Or we can go further and suggest querying, transformation, merging etc facilities. This is an open question for now
PRQ14	Spelling checking		AAAI, QM, MM	Use the thesaurus stored in the semantic layer	Need to decide the best way to implement Language Vocabularies (consider opensearch.org)
PRQ15	Query suggestion		QM, AAAI, MM	Suggest metadata values 'similar' to xx from attributes stored in CERIf entities	similar searches to the ones the user places
PRQ16	Query filter		AAAI, QM, MM	NA	
PRQ17	Datasets viewing	PRQ18	AAAI, AM, IM		we focus only on metadata, not the actual data.We could offer an Amazon like interface, where the different facets (topics or subjects) of the data available are presented

PRQ18	Datasets pre-selection		AAAI, UM, MM		Only authorized-users can set up the list of datasets available in a department
PRQ19	Dataset customization		AAAI, UM, MM		The user (or administrator) narrows the search to specific types of datasets only (notice that we only focus on metadata here). This is different from pre-selection,as preselection is an one-time process, whereas customization is specific to each query.
PRQ20	Linking external resources		QM, AAAI, MM, IM		datasets given by sources not officially connected to the VRE, as long as they satisfy certain baseline requirements
PRQ21	Data Assimilation		AAAI, WM, AM, MM, IM(DMM)		
PRQ22	Data Analysis		AAAI, AM, MM,, IM(DMM), WM		
PRQ23	Data Mining		AAAI, AM, MM, IM		
PRQ24	Data Extraction		AAAI, AM, MM, IM		
PRQ25	Scientific Modelling and Simulation		AAAI, AM, MM, IM(DMM), WM		
PRQ26	(Scientific) Workflow Enactment		AAAI, WM, AM, MM, IM(DMM)		
PRQ27	(Scientific) Visualisation		AAAI, AM, IM		
PRQ28	Data Processing Control		AAAI, WM, AM, MM, IM(DMM)		
PRQ29	Data Processing Monitoring		AAAI, WM, AM, MM, IM		
PRQ30	API		AAAI, IM		

PRQ31	Dataset Download		AAAI, IM		
PRQ32	Quality Rating		AAAI, MM		
PRQ33	Peer Review		AAAI, MM, AM		
PRQ34	Data tag		AAAI, MM		
PRQ35	Data Backup		AAAI, IM		eRIs responsibility

Data Optimization

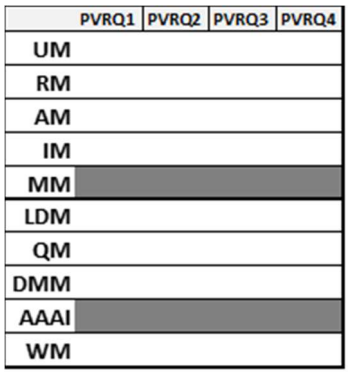


UM - User Manager, RM - Resource Manager, AM - App Manager, IM - Interoperability Manager, MM - Metadata Manager, LDM - Linked Data Manager, QM - Query

Table 3.5 Data Optimization

Req. ID	Description	Rel. with other req.	Components involved	CERIF entities	Notes and Comments
ORQ1	Large datasets processing		AAAI, RM, AM, IM, WM		
ORQ2	Processing parallelisation		AAAI, RM, AM, IM, WM		
ORQ3	Real time processing		AAAI, IM(Adapter), WM		
ORQ4	Data Compartmentalization		AAAI, RM, AM, IM, WM, MM		

Data Provenance

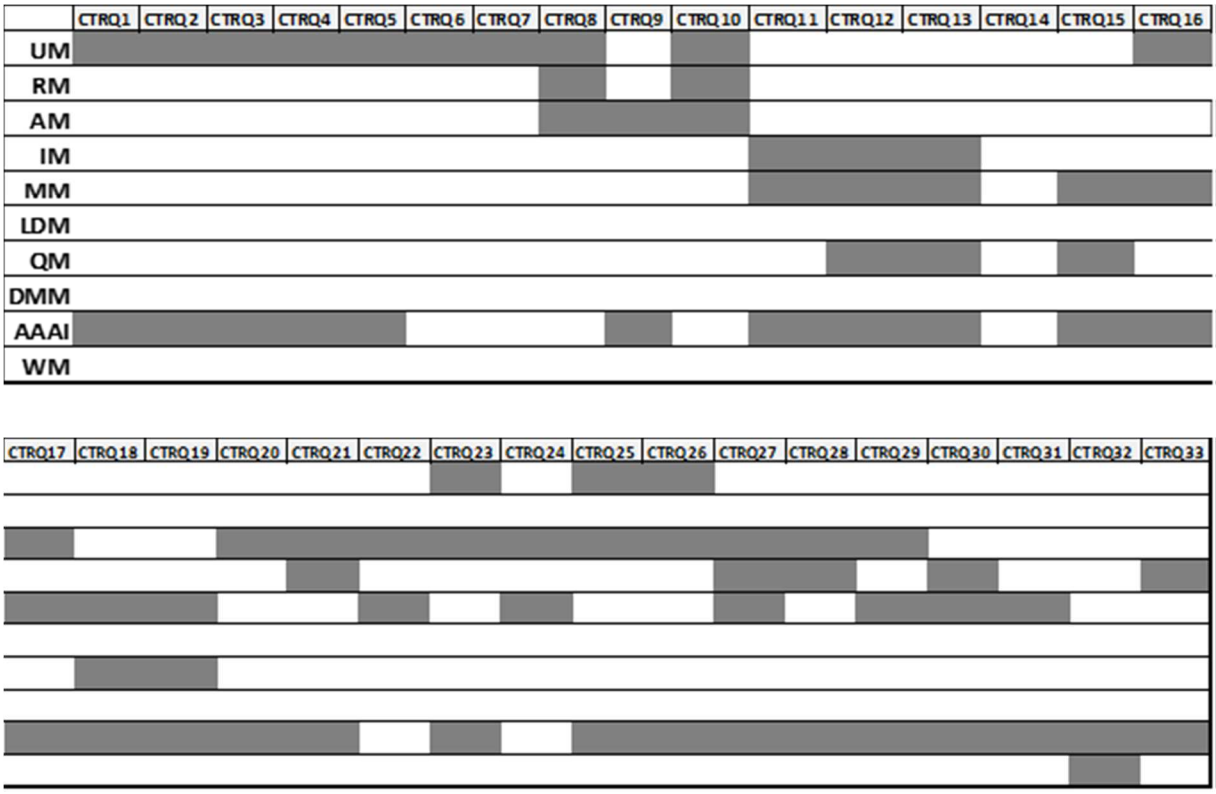


UM - User Manager, RM - Resource Manager, AM - App Manager, IM - Interoperability Manager, MM - Metadata Manager, LDM - Linked Data Manager, QM - Query

Table 3.6 Data Provenance

Req. ID	Description	Rel. with other req.	Components involved	CERIF entities	Notes and Comments
PVRQ1	Data Provenance		AAAI, MM		
PVRQ2	Data acquisition information		AAAI, MM		
PVRQ3	Data curation information		AAAI, MM		
PVRQ4	Data publication information		AAAI, MM		

Collaboration, Training and Support



UM - User Manager, RM - Resource Manager, AM - App Manager, IM - Interoperability Manager, MM - Metadata Manager, LDM - Linked Data Manager, QM - Query

Table 3.7 Collaboration, Training and Support

Req. ID	Description	Rel. with other req.	Components involved	CERIF entities	Notes and Comments
CTRQ1	Login		UM, AAAI	cfPerson for user + some profile information as role in cfPers_Srv, cfPers_Lang	CTRQ2 and CTRQ3 seem special cases of CTRQ1
CTRQ2	Continuous access	CTRQ3	UM, AAAI	Keep information linked to cfPerson (datasets, services,...) with timestamp?	Ubiquity, availability (both non-functional requirements) and multi-channel (functional). Not across devices, but stay connected forever.For example, if connection fails, the session should be the same after we reconnect.

					This means we need to maintain in our servers all users' sessions (scalability issues)
CTRQ3	Single login	CTRQ1	UM, AAAI	cfPerson for user +profile informatio n	Aka Single sign-on
CTRQ4	Interface customizatio n		UM, AAAI	cfPerson for user + profile informatio n	Customize the UI of each user, based on what activities she frequently performs. Customization, localization and personalization, Accessibility is also important
CTRQ5	Wizard configuration	GRQ4	UM, AAAI	cfPerson for user + profile informatio n	
CTRQ6	User instruction		UM	cfPerson for user + cfPers_Me dium, cfPers_Res Publ (doc)	This aspect will include not only tutorials at the VRE4EIC level, but also at the eRI level. This will give incentives for the researchers to generate and upload their own tutorials
CTRQ7	Multilingual interface		UM	cfLang / multilingu al attributes of all entities	CTRQ7 could be included in CTRQ5
CTRQ8	Update alert	CTRQ10	SM(UM,...)		But "activity streams" is much more potent.And it is important for interoperability
CTRQ9	Online dataset editing		AM, AAAI	Link to cfFacility (the RI) that has the cfResProd (dataset),	The tool has to be provided by some eRI and the users only access the datasets of that eRI. We send the VRE users there.

				or to a given cfServ (service)	
CTRQ10	Notification	CTRQ8	SM(UM,...)	cfPerson for user + profile information	There is a subscription mechanism. Consider "activity streams" for this
CTRQ11	Additional services interfaces		AAAI, MM, IM	cfServ	A use case for the VRE system administrator.
CTRQ12	Search for funding	Similar to: PRQ10, PRQ11, PRQ13	QM, AAAI, MM, IM		We assume there is a person populating our catalogue with metadata about project calls. In this case, calls are a resource as any other.
CTRQ13	Funding proposal	Similar to CTRQ12	AAAI, MM, QM, IM		Only the proposals accessible by the authors and the successful ones
CTRQ14	Electronic funding bid				This is irrelevant for us! There are already complex systems for this
CTRQ15	Funding body information		QM,AAAI , MM		Information about name and address of the body or more complex, such as policies? All probably, even though is difficult to implement
CTRQ16	Funding alert	Similar to: CTRQ8, CTRQ10	AAAI, UM, MM		A standard query executed periodically
CTRQ17	Research team setup		AAAI, MM, AM		
CTRQ18	Finding collaborators	Similar to: PRQ10, PRQ11, PRQ13	AAAI, QM, MM		
CTRQ19	Expertise finding	Similar to: PRQ10, PRQ11,	AAAI, QM, MM		These requirements are not only simple queries, they are like recommendation systems. Decision: no recommendation

		PRQ13, related to CTRQ21			
CTRQ20	Forum tool		AAAI,AM		just a simple message exchange system suffices at this point, plug a tool for this
CTRQ21	SNS integration		AAAI, AM, IM		There are simple APIs we can integrate in our system that give twitter buttons, etc
CTRQ22	Group newsletter		AM, MM		
CTRQ23	Meeting organizer		AAAI, AM, UM		Integrate a tool
CTRQ24	Digest email		AM, MM		
CTRQ25	Teleconferencing		AAAI, AM, UM		Considering the integration with a teleconferencing suite
CTRQ26	Instant message		AAAI, AM, UM		Integrate a tool
CTRQ27	Project monitoring		AAAI, AM, IM, MM		A tracking/ticketing system for the project, like dates, deliverables,etc. Integrate a tool
CTRQ28	Computing resource connection		AM, AAAI, IM		
CTRQ29	Education support	CTRQ6	AM,MM,AAAI		Does this require integration with MOOC platforms (in the architecture)?
CTRQ30	Financial information		AAAI, IM, MM		A billing component is needed. Not everyone could use for free certain components
CTRQ31	Accounting		AAAI, MM		
CTRQ32	Workflow engine		AAAI, WM		
CTRQ33	API		AAAI, IM		