SRC | Net

SKAO Regional Centre Network

SKA Regional Centre Network (SRCNet) Software Architecture

SRC-0000001	Revision 02
Classification:	UNRESTRICTED
Document type:	DDR
Date:	2023-09-27
Status:	RELEASED
Authors:	Salgado, Jesús; Wicenec, Andreas; Goliath, Sharon; Joshi, Rohini; Swinbank, John; Bolton, Rosie; Webster, Brendan; Oonk, Johannes; Grainge, Keith; Sánchez, Susana; Parra, Manuel; Dack, Thomas; Hardcastle, Martin; Barbosa, Domingos; Llopis, Pablo; Fabbro, Sébastien; Beswick, Robert; Villote, Jean-Pierre; Breen, Shari; Yates, Jeremy; Grange, Yan; Gaudet, Séverin; An, Tao; Possenti, Andrea; Darriba, Laura; Holanda, Victor; Mendoza, Mariangeles; Galluzzi, Vincenzo; Svedberg, Thomas; Lee-Waddell, Karen; Vitlacil, Dejan; Pandey, Vishambhar Nath; Akahori, Takuya; Chisholm, Louise; Horton, Maya; Watson, Robert;

Role	*Name*	*Designation*	*Affiliation*	*Signature*	*Date*
Author	J. Salgado	SRCNet Software Architect	SKAO	Jesús Salgado	2023-10-03
Owner & Approver	M. van Haarlem	SRC Steering Committee Chair	ASTRON	Michiel van Haarlem	2023-10-03
Released by	L. Ball	SKAO Director of Operations	SKAO	Lewis T Ball.	2023-10-05

TABLE OF CONTENTS

Status of the Document **7**

1. Introduction **7**

1.1 Scope 7

1.2 Overview 7

1.3 The SKA Regional Centres Network 8

1.3.1 The Square Kilometer Array Observatory (SKAO) 9

1.3.2 SKAO Global HeadQuarters 13

1.3.3 Science Users 13

1.3.4 SRCNet Site Developers 14

1.3.5 Multi-Wavelength and Multi Messenger Facilities 15

1.3.6 SRC Sites 16

1.3.7 SRCNet Operations Group 17

1.4 Federation Considerations 17

1.4.1 The SRCNet data lake and the SRCNet nodes 17

1.4.2 The SRCNet and the SRCNet nodes architecture 18

2. Architectural Representation **19**

3. Architectural Goals and Constraints **20**

4. Use-Case View **24**

4.1. Actors 24

4.2 Use Cases Classification 25

4.3 Assessment of architecture using Use Cases 27

5. Modules View **27**

5.1. Architecture Overview 27

5.2. Architecturally Significant Design Packages 30

5.2.1 Presentation Tier 30

5.2.1.1 User Interface 30

5.2.1.2 Interactive Analysis 31

5.2.1.3 System Administration 31

5.2.2 Application Tier API 31

5.2.2.1 Authentication and Authorisation 31

5.2.2.2 Public API 32

5.2.2.3 Management API 32

5.2.3 Application Tier 32

5.2.3.1 System Events and Notifications 32

5.2.3.2 Monitoring System 32

5.2.3.3 Provenance Management 32

5.2.3.4 Data Management 33

5.2.3.5 Metadata Management 33

5.2.3.6 User and Group Management 33

5.2.3.7 Workflow Management 33
5.2.3.8 Resource Management 33
5.2.4 Resources Tier 34
5.2.4.1 Databases 34
5.2.4.2 Execution Framework 34
5.2.4.3 (Data) Repositories 34
5.3 Architecturally Significant Design Packages: 2nd level decomposition 34
5.3.1 Sub-Package Descriptions 36
5.3.2 Services View 48
5.3.2.1 Data Management Service 49
5.3.2.2 Metadata Management Service 49
5.3.2.3 Workflow Management Service 50
5.3.2.4 Authentication and Authorisation Service 50
5.3.2.5 Services Discovery Service 51
6. Process View 52
6.1. Ingestion of data into SRCNet 53
6.2. Data access during processing SRCNet 54
6.2.1. Data staging approach 55
6.2.2. Data mesh approach 56
6.2.3. Data staging vs Data mesh 58
6.3. Science Platform User Interface pattern 59
6.4. Metadata Management System 61
6.5.1 Metadata Replication 64
6.5. Authentication, Authorisation and Accounting 65
6.5.1 AA Interface possible approaches 66
6.6. Collaborative environment 68
6.6.1. User storage areas 68
6.6.2 Persistent Table Upload 70
6.7. Computational resources allocation and federated execution 71
6.7.1. Shared Execution planner 74
6.8. Distributed software management 76
Appendix 1: Requirements Supporting Architecture Principles 77
Appendix 2: Actors fine-grained classification 86
A2.1 Developer role fine-grain classification 86
A2.2 Scientist role fine-grain classification 87
A2.3 Operator role fine-grain classification 88
References 90

LIST OF FIGURES

- Figure 1. SRCNet Software Operating Context
- Figure 2. The ObsProject Lifecycle
- Figure 3. Extension of Fig.2
- Figure 4. SRCNet node blueprint conceptual representation.
- Figure 5. Software Architecture: The "4+1 View" Model.
- Figure 6. First package decomposition representation of the SRC node blueprint as UML packages.
- Figure 7. Condensed SRC node architecture footprint, including layers.
- Figure 8. Sub-packages decomposition.
- Figure 9. Services View
- Figure 10. Data ingestion from telescopes locations into the SRCNet.
- Figure 11. Data access following the “create local copy” approach.
- Figure 12. Data access using the data mesh paradigm.
- Figure 13. Gateway pattern to be used on the client side.
- Figure 14. Metadata management system architectural diagram.
- Figure 15. AA Interface using a global proxy.
- Figure 16. AA Interface not using a global proxy.
- Figure 17. Diagram showing the access from data of one User Storage Area.
- Figure 18. Diagram showing the federated execution of one process.

LIST OF ABBREVIATIONS

AAI	Authentication and Authorisation Infrastructure
ADP	Advanced Data Product
API	Application Programming Interface
AusSRC	Australian SKA Regional Centre
ICD	Interface Control Document
IVOA	International Virtual Observatory Alliance
KSP	Key Science Project
ODP	Observatory Data Product
PID	Persistent Identifier
PLDP	Project-Level Data Product
SDP	Science Data Processor
SKA	Square Kilometre Array
SRC	SKA Regional Centre
SRCNet	Network of SKA Regional Centres
SRCSC	SRC Steering Committee
TAP	Table Access Protocol
UID	Unique Identifier
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WAN	Wide Area Network

Status of the Document

This document has been produced by the SRNet Architecture Forum. It has been reviewed by SRCNet Members and the SRC Steering Committee and has been endorsed by the SKAO an SRCNet formal document. It is a stable document and may be used as reference material or cited as a normative reference from another document.

1. Introduction

1.1 Scope

This document is a technical deliverable describing the SKA Regional Centres (SRCs) Network (SRCNet) software architecture.

This description covers the use cases to be implemented on the SRCNet, the common modules needed for the SRC blueprint implementation, the interfaces between these modules and other SRCNet nodes and the constraints on the implementation. That involves

- Classification of use case types and actors.
- Logical view of the modules and decomposition in submodules of the SRCNet.
- UML diagrams including class diagrams.
- Dynamic aspects of the system, explains the system processes and how they communicate and focused on the run-time behaviour of the system. The process view addresses concurrency, distribution, integrator, performance, scalability, etc.
- UML diagrams to represent the process view include the sequence diagram, communication diagram and activity diagram.

This document does not cover the following items (among others):

- Full compilation of Use Cases from the scientific community.
- Operations Plan for the SRCNet.
- Software Stack to be used for the SRCNet implementation.
- Deployment Plan of the SRCNet nodes.
- Detailed ICD descriptions (e.g. SDP-SRCNet interface).

Due to the development status of the different elements, it would require further updates and review on future versions.

1.2 Overview

Present document is organized as follows:

- Needs of the SRCNet deployment
- Use Cases to guide the architecture design
- Architectural Representation

1.3 The SKA Regional Centres Network

The operating environment of the SRCNet Software is presented in Figure 1. SRCNet Software Operating Context.

The figure describes the most relevant interactions and interfaces:

- *SKAO: Interactions on data logistics of data produced by the telescopes.*
  - As described in section 1.3.1 The Square Kilometer Array Observatory (SKAO)
- *SKAO Global HeadQuarters: Global and operations coordination and proposals information.*
  - As described in section 1.3.2 SKAO Global HeadQuarters
- *Science Users: Interactions of the scientific community with the SRCNet platform.*
  - As described in section 1.3.3 Science Users
- *SRCNet Site Developers: Development teams composed of members of the SRC Sites, coordinated to create the relevant SRCNet software modules.*
  - As described in section 1.3.4 SRCNet Site Developers
- *Multi-Wavelength and Multi Messenger Facilities: Implementation of standards to allow science use cases that imply data from different missions and observatories.*
  - As described in section 1.3.5 Multi-Wavelength and Multi Messenger Facilities
- *SRC Nodes: The SRCNet will be composed of SRCNet nodes. These SRCNet nodes are software instances that will communicate between them using agreed federated protocols and creating a distributed science platform. SRCNet nodes are the basic element of the SRCNet and its architecture is described in the present document.*
- *SRC Sites: The SRCNet nodes will be running in SRC Sites. These SRC sites are physical locations, usually data centres, distributed within the SRC country members.*
  - As described in section 1.3.6 SRC SItes
- *SRCNet Operations Group: Team in charge of the operations of the SRCNet nodes, maintenance and monitoring.*
  - As described in section 1.3.7 SRCNet Operations Group

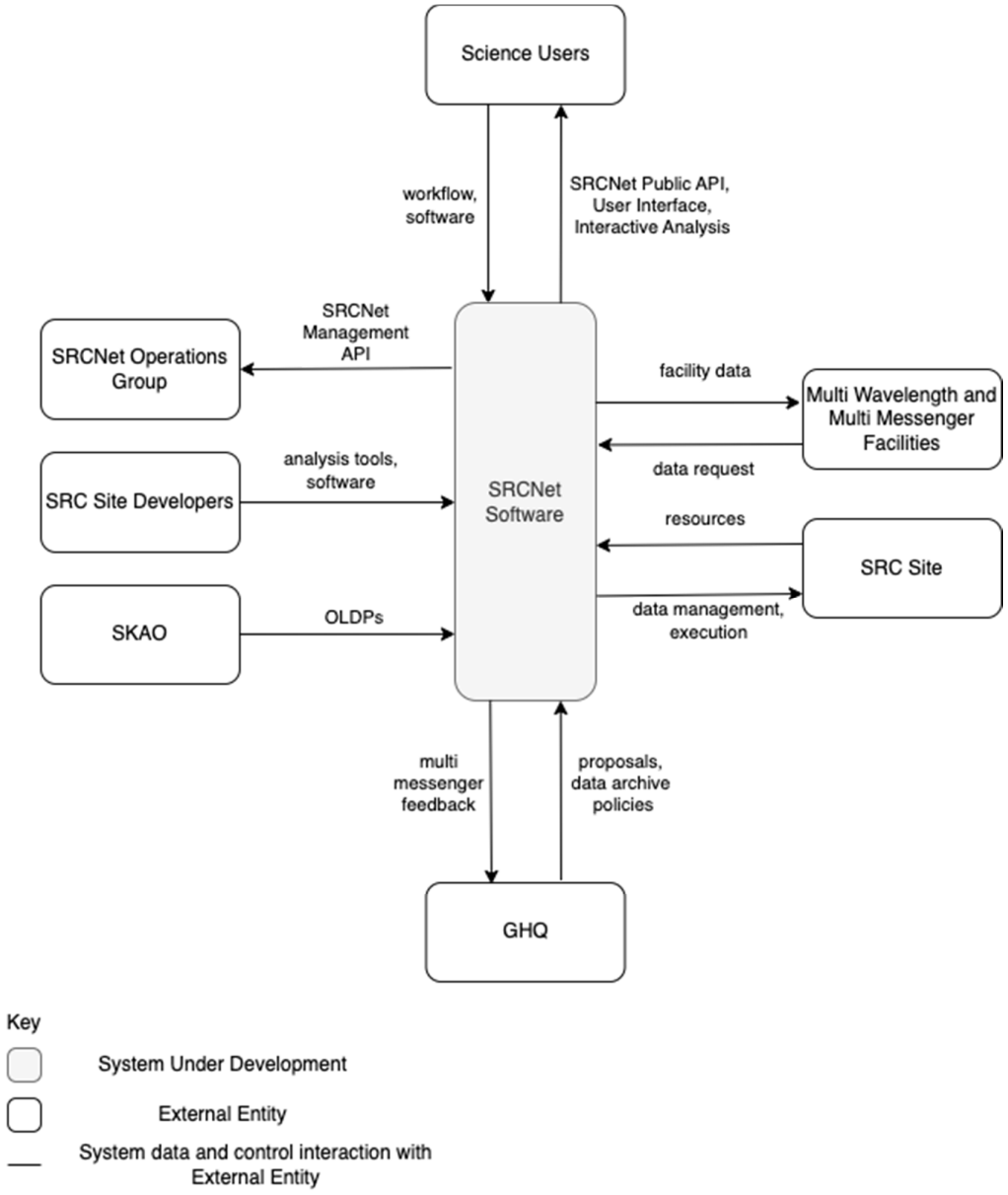

Figure 1. SRCNet Software Operating Context

Software Operating Context of the SRCNet, describes the interfaces to Science Users, external facilities, SRC Sites, SKAO project, development teams and operations team.

1.3.1 The Square Kilometre Array Observatory (SKAO)

The SKA Observatory is a next-generation radio astronomy facility that will help to revolutionise our understanding of the Universe and the laws of fundamental physics. The observatory has three locations: in South Africa's Karoo region (SKA_MID), Western Australia's Murchison Shire (SKA_LOW) and the Global Headquarters in the United Kingdom. The SKA_MID and SKA_LOW locations will be capable of producing a stream of science data products on the order of 700 PB/year. This large data volume is unprecedented

for the astronomical community and thus poses unique challenges for curating and providing access to the datasets and resources required to analyse them in order to derive the final scientific insights. The approach chosen is the development and adoption of the SKA regional centre concept in the form of a loose SRCNet association consisting of, presently regionally funded, regional activities overseen by an SRC Steering Committee. The justification for establishing regional centres is mainly driven by the needs of survey science teams and individual scientists to derive scientific insights from the data products produced by the SKAO (hereafter referred to as Observatory Data Products, ODPs). The very high-level functions required to operate an observatory and derive the anticipated scientific insights are shown in Figure 2. The ObsProject Lifecycle (Quinn et al. 2015). The whole Observatory Proposal lifecycle is segmented into four quadrants. In clockwise order, these are the *Proposal Domain,* the *Observatory Domain*, the *Processing Domain* and the *Science Domain*. The first two cover the proposal and observation phases while the latter two cover the data processing and research stages. Although these domains overlap, they still delineate distinct activities, which can be assigned to different sub-systems and groups of people, some inside the SRCNet Software system.

The SRCNet Software system will provide a scientific data repository for this data using the storage resources available from the SRC Sites (see 1.3.6 SRC SItes). The data rate flow from the telescopes to the SRCNet can be summarised as follows:

- SKA_LOW will produce around 2 Pb/s which will be reduced to around 7-8 Tb/s that will be redirected to the Central Signal Processors.
- SKA_MID will produce around 20 Tb/s that will be redirected to the Central Signal Processors.
- Both Central Signal Processors will produce a combined bandwidth of around 560 GB/s that will be redirected to the Science Data Processor (SDP) for processing.
- The SDP will produce around 700 PB/year, including resilience overheads, that will be distributed to the SRCNet nodes.

All of these data will be distributed into several SRC Sites. SRC Sites will provide access to the SKA data as well as computing resources necessary to fully exploit their science potential.

In order to reduce the initial data volume to a manageable amount, it has been decided that the SKAO would process the data to a significantly higher degree than is typical for current radio facilities. For radio astronomy data that means multi-dimensional *Data Cubes, Catalogues* and a number of more specialised products for specific science projects. The SDP workflows are developed and executed in a very controlled environment using carefully tuned settings, to satisfy a broad range of science goals, with limited adjustability required by the users.

The list of expected OLDPs is identified at (Breen, Bolton, and Chrysostomou 2021).

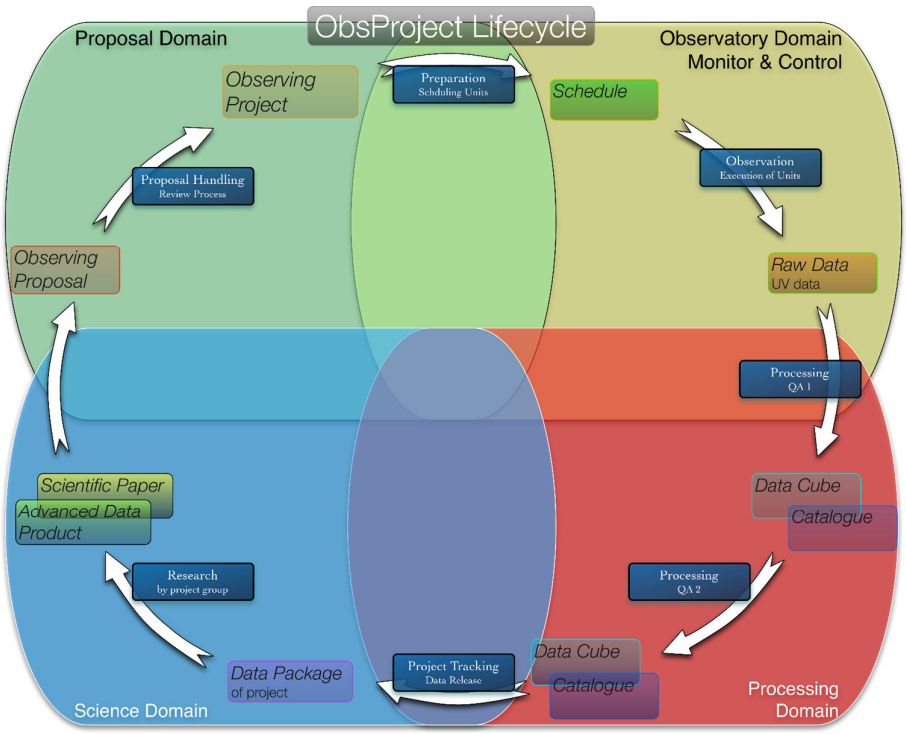

Figure 2. The ObsProject Lifecycle

It starts with scientists preparing Observing Proposals, which will then be turned into actual Observing Projects utilising the Proposal Handling process. The set of accepted Observing Projects is then turned into a Schedule which is then executed to carry out astronomical observations which results in Raw Data also termed as UV Data. In radio astronomy, this is a multi-staged, quite complex process, even to just arrive at the UV Data indicated in the figure. It is followed by some processing, also to be able to decide whether the collected raw data meets certain basic criteria (QA1). In order to reduce the initial data volume to a manageable amount, it has been decided that the observatory would go further and process the data to a significantly higher degree. For radio astronomy data that means in general multi-dimensional Data Cubes, Catalogues and a number of more specialised products for specific science projects. Again, to assess the quality of those products and thus the success of the whole observation there is a second quality assessment (QA2) built-in. After that typically all observations belonging to a certain Observing Project (unless there is further processing to generate project-level data products) will be collected, packaged and released. The released Data Package(s) can then be picked up by the science teams and further analysis. This step constitutes the actual Research activities and results in Advanced Data Products and Scientific Papers. Very often these results lead to follow-up Observing Proposals and thus closes the cycle.

Users of the SKA telescopes be able to apply for time and resources under three project types: Key Science Projects (KSPs), Principle Investigator (PI) Projects and Director General's Discretionary Time (DDT) proposals. KSPs are expected to require significant observing time and resources and may make up as much as 70% of the observing schedule in full operations, putting their requirements at the forefront.

The SDP will produce Observation-level data products (OLDPs) following the conclusion of each observation and queue them for delivery to the SRCNet once QA has been passed. In many cases, the scientific user will have requested Observatory Data Products that are the combination of a number of OLDPs, called Project-level data products (PLDPs). While these PLDPs will likely be created within the SRCNet Software System, the pipelines to produce them, including the provision of associated provenance and QA information,

remain the responsibility of SKAO. Both OLDPs and PLDPs will have passed rigorous QA before they are released to the proposers, removing the SRCNet Software system from a time-critical processing and decision loop since the quality assessment processing and final approval will completely reside within the SKAO. Any re-scheduling of observations (yellow arrows in Figure 3. Extension of Fig.2) due to insufficient quality would then also be at the discretion of the observatory.

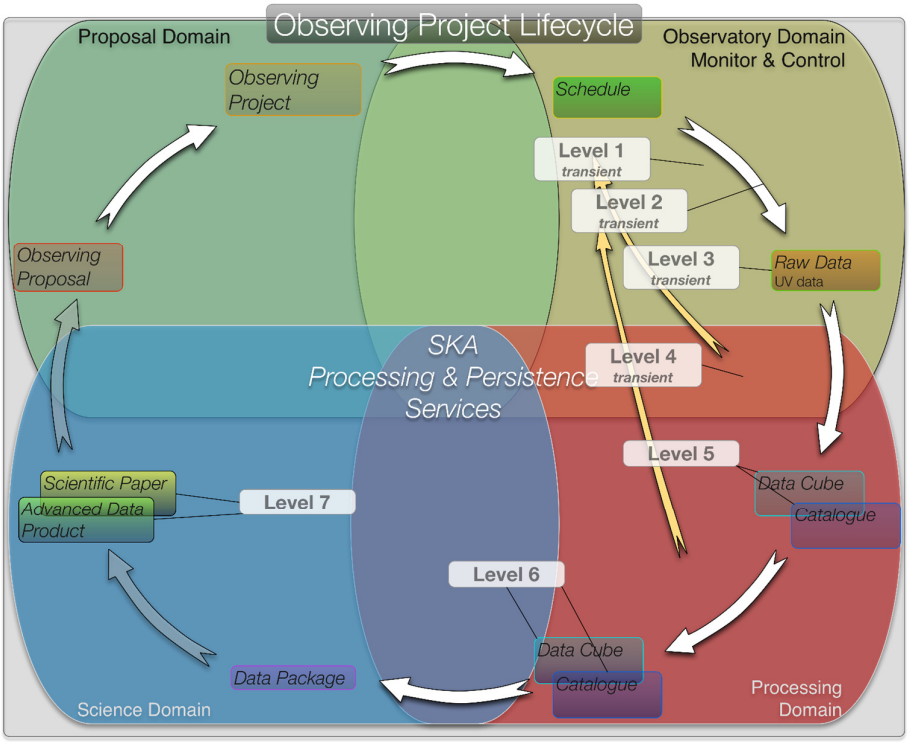

Figure 3. Extension of Fig.2

including the various levels of data reduction as well as the generic and abstract SKA Processing & Persistence Services. These services also include the actual processing frameworks, workflows and algorithms as well as the SKA Science Archive. The term Persistence Services is chosen here to indicate that there is quite a bit more required for the overall SKA than just the Science Archive to ensure that the data is managed and curated appropriately throughout the lifecycle and indeed over the lifetime of the observatory and potentially beyond. The Level 1-4 data products are labelled transient, which means they won't be stored longer term. The Level 5 and Level 6 data products essentially are the Observatory Data Products (ODPs), where Level 5 is before the final quality assessment and Level 6 is after. This is a critical distinction since the only ODPs worthwhile keeping long-term and further analysing are the Level 6 ones. Level 7 products are derived products, often involving data from other observatories or experiments as well as scientific papers and other associated products.

The SKAO coordinates the detailed proposal selection, preparation, execution of the observations, initial calibration and generation of the ODPs adhering to a pre-defined quality. After that, the SRCNet Software system takes over again and supports all the additional processing required to produce the ADPs.

Both the ODPs and the ADPs will need to be curated and maintained by the SRCNet in a FAIR (findable, accessible, interoperable, and reusable) and secure way.

After QA2 the SRCNet Software will collect, pack together, and release all observations belonging to a certain *Observing Project*, along with applicable calibration and auxiliary data and logs. This packaging is performed by the SRCNet System software.

The interface with the SKAO is described in the following reference document (Diamond and Quinn 2020).

1.3.2 SKAO Global HeadQuarters

The Global HeadQuarters (GHQ) is the third SKA site. For the interface to this site, the SRCNet Software System will implement the governance policies according to the technical oversight of the Global HeadQuarters.

The GHQ coordinates the detailed proposal selection and preparation and execution of the observation.

The SRCNet will have some responsibilities to support users in the preparation of proposals (*Proposal Domain in Fig 3)* and likely also in the proposal review process as well as user support during the execution of a project.

The Level 1-4 data products are labelled *transient*, which means the SRCNet Software System will not store them long-term. The Level 5 and Level 6 data products essentially are the Observatory Data Products (ODPs), where Level 5 is before the final quality assessment and Level 6 is after. The SRCNet Software System will keep the Level 6 OLDPs and the Level 7 ADPs long-term (reference for Level 1-7 descriptions here), in accordance with the GHQ data archive policies in this applicable document (reference TBD).

The governance interfaces with the GHQ are described in this reference document (Diamond and Quinn 2020).

1.3.3 Science Users

The SRCNet Software System will provide, in collaboration with the SKAO, science user support across the entire Observing Project lifecycle as described in Figure 3. Extension of Fig.2, from initial discussions around the capabilities and limitations of the SKA telescopes to the final delivery and curation of the ADPs. This means that the SRCNet will share some responsibilities in the *Proposal Domain* as well as in the *Processing Domain*.

The SRCNet Software System will enable and support scientists and science teams to analyse the ODPs delivered by the SKAO. Some of the methods and workflows envisioned to perform this analysis will be novel, or at least non-standard and will require significant effort to make them applicable to SKA-scale data analysis. Meeting this challenge requires a far tighter collaboration between the technical experts, the software developers and the scientists than is typically necessary. In order to first enable an analysis of what that scale is and what the actual challenges are, we can draw on work done in the past. (Quinn, 2015) already presented an initial insight into the expected data rates and volumes for a whole set of Key Science Projects (KSPs).

The SRCNet Software system workflows will provide a flexible setup, allowing scientists to develop their own workflows and components and also provide some experimentation capability.

The SRCNet Software System will release *Data Package(s)* that can then be picked up by the science teams for further analysis.

The interface with Science Users is described in Sections 5.2.1 Presentation Tier and 5.2.2 Application Tier API of this document.

1.3.4 SRCNet Site Developers

The SRCNet Software System will *assist the SRC Sites in providing access to the software that implements the science use cases of the community, including those of the large surveys.* The SRCNet Software system will provide these projects with two things:

- a generic infrastructure and environment capable of handling SKA-scale data and processing rates and volumes and
- dedicated support for each of the KSPs, but in particular the extremely large ones.

SRCNet Site Developers will provide specific science analysis capabilities, software and tools, required to achieve the science goals set out by each of those projects to the SRCNet Software system. In order to support such a multitude of different scientific reduction workflows on large-scale computing platforms and involving TB and PB size datasets requires an emphasis on the way workflows and their algorithmic components are developed and maintained.

The SRCNet Software will provide access to a federated development, testing, and execution environment for these analysis tools and software.

In addition, a close collaboration with the SKA Science Working Groups will be needed to ensure that the use cases and requirements, and thus the analysis tools and software being developed, are kept up to date with the ambitions and plans of the science community.

The SRCNet will facilitate the addition and removal of SRCNet Site Developers, as funding might come and go and the availability of SRCNet Site Developers dedicated to SRCNet Software system activities can't be assured.

The SRCNet Software system implementation is a common effort that implies the creation of SRCNet Site Developers distributed all across the planet and the creation of synchronised development cycles.

Development resources will be provided by the different SRCNet partners and they will work in coordination to develop all needed components of the SRCNet Software system. It is expected to provide a gradual increment of functionality and capacity until being fully functional and with full capacity for the SKA assembly AA* (Salgado 2023).

The interface with SRC Site Developers is described in the reference document (Diamond and Quinn 2020).

1.3.5 Multi-Wavelength and Multi Messenger Facilities

The SRCNet Software System will support Science Users with the merging of the SKAO data and data from other wavelengths to assist in the production of scientific insights.

Modern astronomy is multi-wavelength and multi-messenger (MM, hereafter) and thus the vast majority of the SKA science projects will require the usage of data from other facilities. In fact, synergies between the major planned and operational facilities have been established already and are being actively explored (e.g. Euclid[1], Vera Rubin[2], LIGO[3], ESO ELT[4], JWST[5], NGVLA[6]). For some science projects the SKA data is the add-on, rather than the driver and some of the SRCNet nodes will potentially may even host data for multiple of these facilities as service providers. Such MM science projects typically have a very high impact and provide unique insights into the physical properties of the studied astronomical phenomenon. Performing such research adds a lot of complexity to the SRCNet in the sense that multiple of these data sets have to be understood and merged, but there might also be a competing demand by MM studies with other facilities. On the more technical side supporting MM projects essentially makes the usage of IVOA protocols and standards mandatory. Some of these MM projects (e.g. GW and GRB follow-ups) will require quick turn-around times and thus resources need to be made available on short notice by the SRCNet.

For some of the transient MM projects, the SRCNet Software system will need to support quick follow-up observations on the SKA telescopes, associated with the required data reduction to derive results in a timely manner. General prioritisation plan of data processing tasks will be defined in the operations plan and it will require a case-by-case study.

Some of the interfaces with Multi-Wavelength and Multi-Messenger Facilities are described in the documents (Dowler et al. 2019) (Louys et al. 2017) (Salgado and Ibarra 2021) (Allan, Denny, and Swinbank 2017).

1.3.6 SRC Sites

The SRCNet Software System will *provide the software to federate the regional and community sites into a working, usable and maintainable SRCNetwork.*

The SRCNet Sites are completely independent entities, each with its own funding cycles, local government policies and additional responsibilities not related to the SKA. Funding might be granted only in the context of a bigger, local endeavour and thus there is a likelihood of strings attached to it. Funding might

[1] https://www.cosmos.esa.int/web/euclid

[2] https://www.lsst.org/

[3] https://www.ligo.caltech.edu/

[4] https://elt.eso.org/

[5] https://webb.nasa.gov/

[6] https://ngvla.nrao.edu/

come and go, therefore the SRCNet Software system will support the addition and removal of SRCNet Sites.

Another point of critical concern is about the scale of the required resources to support the activities of the science teams and later also the broader community to further analyse the archival data. Typically the teams around a large KSP will consist of ~100 scientists. A single KSP will have one or a few main science drivers which are covered by the discussion in 1.3.3.1. However, the interests of each of the involved scientists, driven by the necessity to produce first-author papers as well as the mere potential of the data at hand, are far more diverse than that. It can thus be assumed that there will be at least ~50 additional science projects proposed by the core science teams. Once open to the broader community, there could potentially be hundreds more. In order to achieve the main objective of the SRCNet to maximise scientific output and impact, the architecture needs to embrace this diversity and try to support as many high-quality science projects as possible while avoiding duplication across the science projects to the maximum extent possible.

As the sole curator of the exabyte-scale SKA Science Archive, the SRCNet Software system will require an additional (or expanded) proposal and review system for science projects solely based on archival data. Given the rough estimates above this will mean that several hundreds of science projects will compete for resources on the SRC sites, **in addition** to the, relatively few, main science projects being observed at any given point in time.

The SRCNet Software system will select and distribute these projects within the SRC Sites. The SRCNet Software system will schedule the load and the relative priority of each of them. The SRCNet Software system will provide tools and systems to support decision-making with a far broader scope than the SKAO. In the most extreme cases, the SRCNet Software system will prioritise a high-impact, very demanding archival project over a newly proposed observational KSP at any number of SRC Sites.

Some of the interfaces with SRC Sites are described in 5.3.2.1 Data Management Service, 5.3.2.2 Metadata Management Service, 6.7. Computational resources allocation and federated execution sections of this document and in (Diamond and Quinn 2020).

1.3.7 SRCNet Operations Group

The SRCNet Software system will make use of dedicated support personnel assigned to them.

The interface with SRC Sites is described in the reference documents (SKAO 2020), (Diamond and Quinn 2020)

1.4 Federation Considerations

1.4.1 The SRCNet data lake and the SRCNet nodes

Generally, data lakes (Gorelik 2019) are centralised repositories designed to store, process, and secure large amounts of structured, semi-structured, and unstructured data. It can store data in its native format and process any variety of it.

However, the SRCNet repository will be centrally managed but distributed and federated at the storage elements level. So two challenges should be addressed:

- Data exploitation of the data lake: Although data lakes provide full access to data for analysis to the users, exploitation of data lakes is complex as the computation is not integrated into the data. The SRCNet should include big data lakes exploitation techniques. Generally speaking, a data lake is a storage repository that contains a vast amount of data, at different levels of processing, that requires integrated analysis applications for general processing, including classical ETL (Extract, Transformation, Load) but, also, more advanced analysis techniques.
- Data Latency due to distributed repositories: Although some use cases would require a preliminary movement of the data to start the analysis, methods should be provided to execute remote operations on the data to minimise data transfers whenever possible.

As a consequence of this distributed computing, the SRCNet will not be composed only of the data lake but, also, of computing nodes to process these data. For the optimal consumption and analysis of the SKA data, the nodes should be able to share executable code and they should be able to share network topology metadata (available resources, network connectivity, occupancy, etc).

1.4.2 The SRCNet and the SRCNet nodes architecture

The present document describes the architecture of the SRC nodes and the SRCNet to allow access to the scientific community to the SKA science data. The modules and interfaces described in this document allow different levels of implementation:

- Basic blueprint: A basic blueprint is an architecture software design that allows proper communication of the SRC nodes and similar access and usage of the SKA data into the SRCNet. The SRCNet should produce an implementation of the modules described in the Architecture blueprint. A basic SRC node would be a deployment of these reference implementation modules at their local SRC.
- Replacement by a different API implementation: Due to local constraints or technical reasons, the reference implementation could be substituted by a different implementation for some of the modules. This is particularly relevant for the resources layer modules as local infrastructure could vary. In all cases, these alternative implementations should preserve the API agreed upon for the module, so the rest of the SRC elements could be integrated without changes.
- Extension of components: Components could be extended in functionality by adding extra features. In this case, the new implementation should be backwards compatible with the basic API described for the architectural blueprint of this module so the new interface should be an extension of the basic one. Other modules that only understand the basic interface should be able to be interconnected to the new component implementation without changes, although not being able to use the extended functionality.

This architectural blueprint for an SRCNet node follows a federated model and it is being defined to facilitate compliance with the FAIR principles (findability, accessibility, interoperability, and reusability) and the use of IVOA standards whenever applicable. Implementation of these principles and the use of

standards allow data exploitation using already existing clients or newly created ones using documented APIs. In this way, the scientific community would be able to connect to any SRC to access SKA data and run analysis tools in the most straightforward possible way, simplifying the technical complexities of using SRCNet services for the scientists and maximising scientific data exploitation.

The blueprint will specify tools to discover SKA data, computing resources, interoperability services, visualisation tools, science-enabling applications, and common SKAO SRCNet support to the scientific community in the form of helpdesk support, training, and project impact dissemination. All SRC nodes will share a common data logistics management system that will manage data distribution, minimise unnecessary duplication, share data following availability criteria, and identify popular data sets to create copies for fast access. Also, all the SRC nodes of the SRCNet will be connected between them using federated services for, among others, a common authentication/authorisation system, federated data access, federated execution, global monitoring and events system, etc.

These considerations indicate a need for a paradigm for managing and deploying the federated services across distinct, heterogenous SRC nodes. The Service Mesh paradigm is important to consider in this respect. At its very core, it provides secure service to service communication and Layer 5 management. This can mean providing service level orchestration and management on top of containerised services, such as those running in Kubernetes, and also traditional non-containerised VM-based services. Service meshes can allow for API-based management of distributed core SRC services, including monitoring, version management, cross-compatibility of these services and traffic control among the distributed instances of running services. This will grant the SRCNet load-balanced and fault-tolerant core services running across the nodes. Support for different types of service implementations will allow for SRC node variety (in terms of resources available, how resources can be provisioned, person power available and local skillsets) while enforcing the respective Service Level Objectives (SLOs).

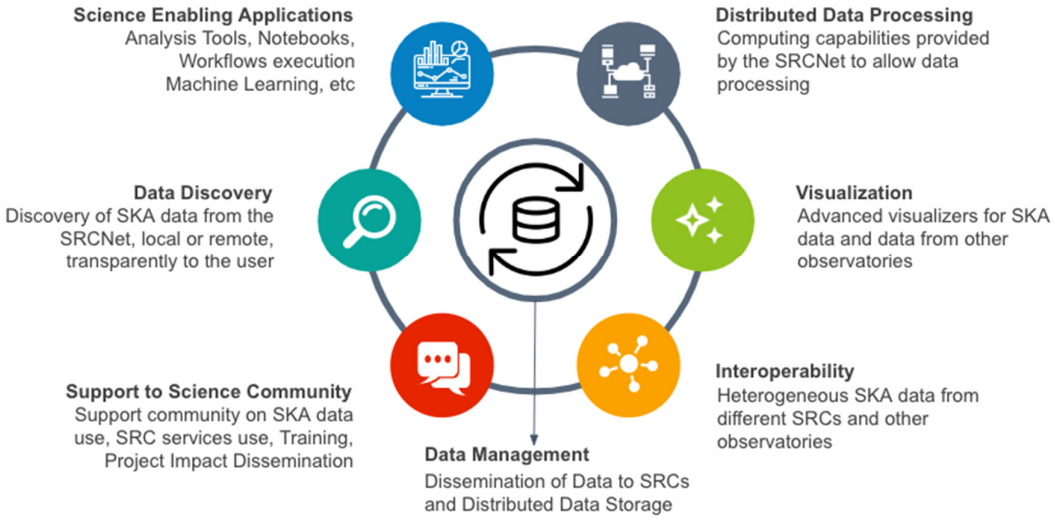

Figure 4. SRCNet node blueprint conceptual representation.

All the SRC blueprint implementations, the data lake, and the protocols involved will compose the SRCNetwork or SRCNet.

However, it is foreseen that this common SRC blueprint can be extended by increasing the capabilities of one particular area or by specialisation in domains. This would allow the creation of expert groups, probably defined by data or science domains, disseminated across the SRCNet to provide better support and more scientific results for the community.

2. Architectural Representation

The software architecture representation will be done following the "4+1 architecture view model"[7]

The "4+1 architecture view model" (see Figure 2 Software Architecture: The "4+1 View"…) illustrates five uniquely viewed perspectives in the design of the architecture: Use-Case, Logical, Process, Deployment, and Implementation views.

The logical architectural representation of the SRCNet architecture will be composed of a Logical View, a detailed modules view representation showing submodules and interrelations, and a detailed module decomposition whenever appropriate.

As the development of the SRCNet is still under design at this stage, the current document architecture representation will be done through a Conceptual Architecture View. That implies that although the modules and the interrelationships between them will be sketched, many of the implementation details will be only present in future versions of the architecture.

The Use-Case View describes functional and non-functional significant system L1 requirements. Use cases are used to generate a consistent set of requirements and to constitute the glue that unifies all the other views.

The present document will cover the following aspects of the 4+1 View:

- Logical view
- Process view

Use Cases (Scenarios) will be covered in a separate reference. The development view and the Physical view will be described in the future Implementation plan.

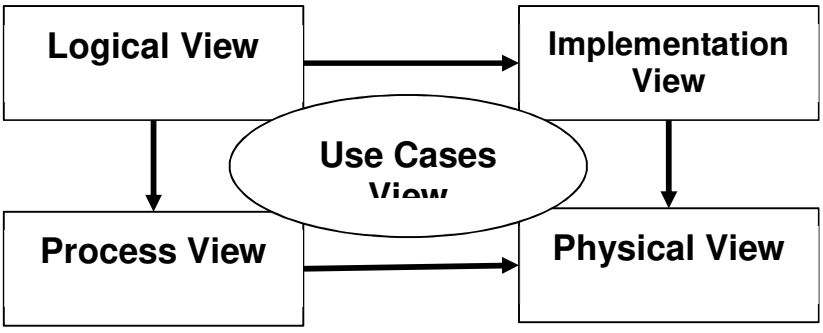

Figure 5. Software Architecture: The "4+1 View" Model.

[7] https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

3. Architectural Goals and Constraints

There are many factors that imply a complex design and implementation of the SRCNet.

The large data volume to be shared and analysed, data distribution into a federated multi-national/multi-continent network, complexity of the algorithms required by the world wide scientific community for proper data analysis coupled with the need to provide distributed intercontinental computational resources make the SRCNet one of the most complex software entities within the context of scientific data processing.

Several groups all across the planet will contribute to implementing this complex distributed platform, with different budgets, expertise and internal organisational rules. Also, different groups of experts on the development of science data platforms and analysis workflows across the network will need to agree on software elements' design decisions, some of which are controversial. Due to this federated pledging approach, consensus could be difficult to reach dealing with different national interests.

This is why a decalogue of rules would help to guide the design and future implementation decisions. These rules would help to obtain the best possible SRCNet software entity in the world.

1. The main objective of the SRCNet is to maximise the science produced by the community using SKA data

The definition of the SRCNet Architecture should be developed in coordination between the scientific and engineering communities to ensure that the design allows producing valuable science from the SKA data.

The measure of the maximisation of the science produced using the SRCNet would be regularly monitored by SKAO using metrics that could include among others:

- Metrics that combine the number of publications produced by the scientific community using the SRCNet and its combined impact.
- Number of use cases implementable using SRCNet components.
- Percentage of the SKA data linked to publications.
- Performance metrics related to scientific productivity.
- Surveys.

Combined metrics will be provided in a verifiable way and comparable among facilities.

The primary goal of maximising the science produced within the SRCNet should be to accommodate the approved science programme, and, in a second level of priority, to allow the execution of science use cases for multi-mission, public data or other kinds of scientific studies. In case of conflicting decisions, the primary goal takes preference.

2. The SRCNet development is a global effort done by all the SRCs

SRCNet is being created by and for the SRCs to ensure a global discovery, access and execution system. SRCs are software providers and consumers. The effort provided by the SRCs, national funders and partners on the development must be properly acknowledged to success in the global effort.

3. The SRCNet architecture should be scalable

SRCNet architecture should allow adding new resources to the net to scale its capacity. This should be allowed in two ways. Locally, nodes should be scalable to new resources. Globally, the SRCNet should allow adding new nodes and SRC partners.

4. The SRCNet architecture should be extensible

Architecture should allow SRCNet nodes capabilities extension. SRCNet node basic capabilities could be extended while remaining backwards compatible. Also, the SRCNet architecture should be able to react and adapt to future changes.

5. The SRCNet architecture should provide data and Computing Resilience

SRCNet architecture should prevent the unavailability of data access and computing resources due to the failure or unscheduled maintenance of a particular node. Also, the nodes should be designed so that disruptions are minimised.

All SRCNet nodes should comply with an agreed set of basic capabilities, which implies a Multi-Region Architecture. Users should be able to transparently access these services. This architecture will maintain high availability in case of regional failover.

6. The SRCNet architecture should follow FAIR principles

SRCNet components should follow FAIR (Findable, Accessible, Interoperable, Reusable) principles whenever applicable. The use of these principles makes digital objects more valuable as they are easier to find through unique identifiers and easier to combine and integrate thanks to the formal shared knowledge representation.

7. The SRCNet Architecture should be designed to minimise cost and environmental impact and maximising throughput

Due to the distributed nature of the SRCNet data lake, the architecture should allow for the "moving code to the data" paradigm as the input data for scientific data reduction pipelines is often larger in volume than the code (including in some cases the operating environment) to run these pipelines. Latency produced by the movement of big amounts of data to an execution SRCNet node from a remote SRC, even using temporary copies, should be prevented as much as possible for big datasets. Also, the creation

of temporary copies of big datasets would increase the cost of the whole SRCNet in terms of storage and network use.

While the location of the largest input datasets is one of the main criteria to decide on which SRC node the pipeline will be executed, there might be scenarios where moving the data to a different node may be more efficient (e.g. when the node hosting the input dataset has not enough resources to allocate the pipeline execution). The architecture should also allow the transfer of input data in case of those scenarios. Also, the architecture should allow the transfer of intermediate and final products of workflow execution across nodes for this kind of pipelines.

In general, the SRCNet Architecture should contain optimisation algorithms to suggest the best possible execution plan for the different pipeline types.

8. The SRCNet architecture should allow federated execution

Jobs could be executed in any SRCNet node, regardless of where the user submitted them. By default, the execution node would be automatically assigned by the SRCNet code based on data and computing resource availability, meaning that there should be enough FLOPs available to process the data stored locally in the online archive (within an acceptable amount of time defined by the non-functional requirements.

To perform this task, a global accounting and monitoring system (that should include, at least, personal storage for users and groups, computing resources used, configurable quotas per users and groups (storage and computing), etc) is needed to track the resources used by the SRCNet users either in local SRCs or globally. Policies on accounting are to be defined and agreed upon.

9. The SRCNet architecture should allow reproducibility of the execution of analysis workflows

Due to the size and complexity of execution of the SKA data-analysis flow, the workflows should be reproducible into the SRCNet. That means, workflows should be serialised in a documented format that includes all the different needed dependencies like input data, execution parameters, software versions, etc. SRCNet implementation should allow replicability (reproduction of executions with identical results) and reproducibility (possibility to execute with varying parameters).

10. The SRCNet architecture should ensure Data Integrity

The SRCNet should ensure the curation of the project digital objects (SKA pipeline data, official software entities, etc) to maintain their value throughout their lifecycle of interest and usefulness and the preservation of these digital objects within the net for as long as required.

11. The SRCNet architecture should be secure

The SRCNet should be designed to be secure in terms of authorisation access rights, use of compute resources, user information and prevention of execution of malicious software. Also, coordinated security rules should be placed for the SRCNet nodes.

12. The SRCNet architecture should minimise the environmental impact

The SRCNet should be designed to minimise the environmental impact on the operations, including efficient use of the resources. A correct balance between environmental impact reduction and the reduction of operational costs should be obtained.

Requirements associated with architectural principles are identified in Appendix 1

4. Use-Case View

A set of documents with the list of use cases is provided by the Science Engagement group on the expected features that the SRCNet would need to implement. The first one is a list of general use cases (Clarke et al.) and the second is a list of visualisation use cases (Franzen et al.). Using this list, the identification of actors and classification of Use Cases will be extracted.

4.1. Actors

By the analysis of the provided Use Cases, we can extrapolate a coarse-grained classification of the roles involved in different use cases.

Role	Description
Developer	Someone who is developing any of the SRCNet components to provide science platform services or infrastructure handling.
Scientist	Someone who is making use of the SRCNet tools and services to search or analyse data.
Operator	Someone who is in charge of deploying and maintaining services, handling incident logs and performing troubleshooting on one or more SRCNet components.

This coarse grain representation can be decomposed on more detailed roles described in the Appendix 2: Actors fine-grained classification

4.2 Use Cases Classification

Although all the use cases present in the documents are quite different, an inspection of them allows classification in terms of analysis, discovery, visualisation and operations areas. These ambits are described as follows:

Use Case Type	Primary Actor	Supporting Actors	Description	Needed Software elements basic decomposition
Access and/or Retrieval Data	Scientist	Developer Operator	A end-user user wants to access or retrieve Data, either using a graphical user interface or a scripting interface Modules to ensure access policy and data transfer to be implemented Monitoring and accounting tasks by Operators tools	Science Archive Data discovery service Data access service Databases Repositories Authentication Authorisation User Interface/command line interface
Interactive Data Analysis	Scientist	Developer Operator	A final user wants to define and execute an algorithm on an interface using an interactive interface and obtaining results on the different steps This interactive algorithm could be converted into an non-interactive one for batch processing	Science Archive Interactive interface Authentication Authorisation Computing allocation Execution framework Software workflows repositories
Algorithm execution	Scientist	Developer Operator	A final user wants to execute analysis on some data, usually through a scripting interface	Interactive interface Authentication Authorisation Computing allocation Execution framework

			Algorithm access and execution modules need to be implemented Federated execution could be needed if data to be analysed is remote and movement of data is a penalty Monitoring of resources and control of the algorithm execution operation by Operators	Software environments repositories Workflows management SRCNet topology
Visualisation	Scientist	Developer Operator	A final user wants to visualise (and, possibly, take actions on the visualisation) a certain data set, usually through a visualisation tool interface Visualisation tools need to be developed and maintained Interfaces to data lake modules need to be developed Server tools if needed, and processes	Visualisation tools Visualisation server modules Service discovery Data discovery service Data access service Databases Repositories Authentication Authorisation
System Operations	Operator	Developer	An operator needs to perform an operation into the system, due to an alert, a maintenance action or an operational procedure. Monitoring tasks by the operations team usually make use of specific user interfaces and dashboards. Operations procedures are	Administrative interface User/Groups management Monitoring service System events system

			performed by invoking specific command line scripts Interfaces and modules should be developed and maintained to perform operations on the network, locally and/or remotely	

4.3 Assessment of architecture using Use Cases

The SRCNet Architecture will follow an assessment process by using selected scenarios based on use cases to ensure alignment. One possible approach to be followed is the Architecture Tradeoff Analysis. The architecture review will provide relevant information to improve the architecture definition and relevant architecture decisions.

5. Modules View

This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages, and for each significant package, its decomposition into classes and class utilities.

Introduce architecturally significant classes and describe their responsibilities and a few important relationships, operations, and attributes.

5.1. Architecture Overview

Although the SRCNet nodes are complex entities that globally create the SRCNet, every node can be decomposed following a traditional simple layer structure. Several packages, and groups following a three-layer architecture, can be identified.

The layers identified and the associated packages are as follows:

- **Presentation Layer:** User and Communication layer accessed by scientists that would connect to the SRCNet nodes application layer. This layer is composed, among other modules, of the User front-end, interactive analysis tools, a system administrator tool, data visualisation, command-line client libraries, etc. Different secured components of the presentation layer could be also provided for operators and administrators. These components could connect to public and private server layer API methods. Due to the international scale of the project, the presentation layer should be implemented following the “Internationalisation and Localisation” pattern. The number of available languages for common services should be significant (translations provided by SRC members for their

own local language of the text terms of the application) although the number of possible languages could be reduced for service extensions.

- **Server/Application Layer:** Interfaces and modules running on the server side. This would be composed of a general API (both public and private for administration), an Event Bus, a monitoring system on the server side to prepare reports, and a set of management packages to handle data, metadata, provenance, user and group profiles, access rights, workflow management, and resource management.
- **Resource Layer:** Collection of resources with stored metadata, data, and procedures. This is composed of data repositories, databases, workflows (i.e. container images, pipelines, and code or hooks for fetching the code) and, in this particular case, also by computing resources available for analysis.

The three-layer architecture allows an easy division of development responsibilities and easier identification and isolation of problems but also helps to improve the scalability, reliability, and security of the final product.

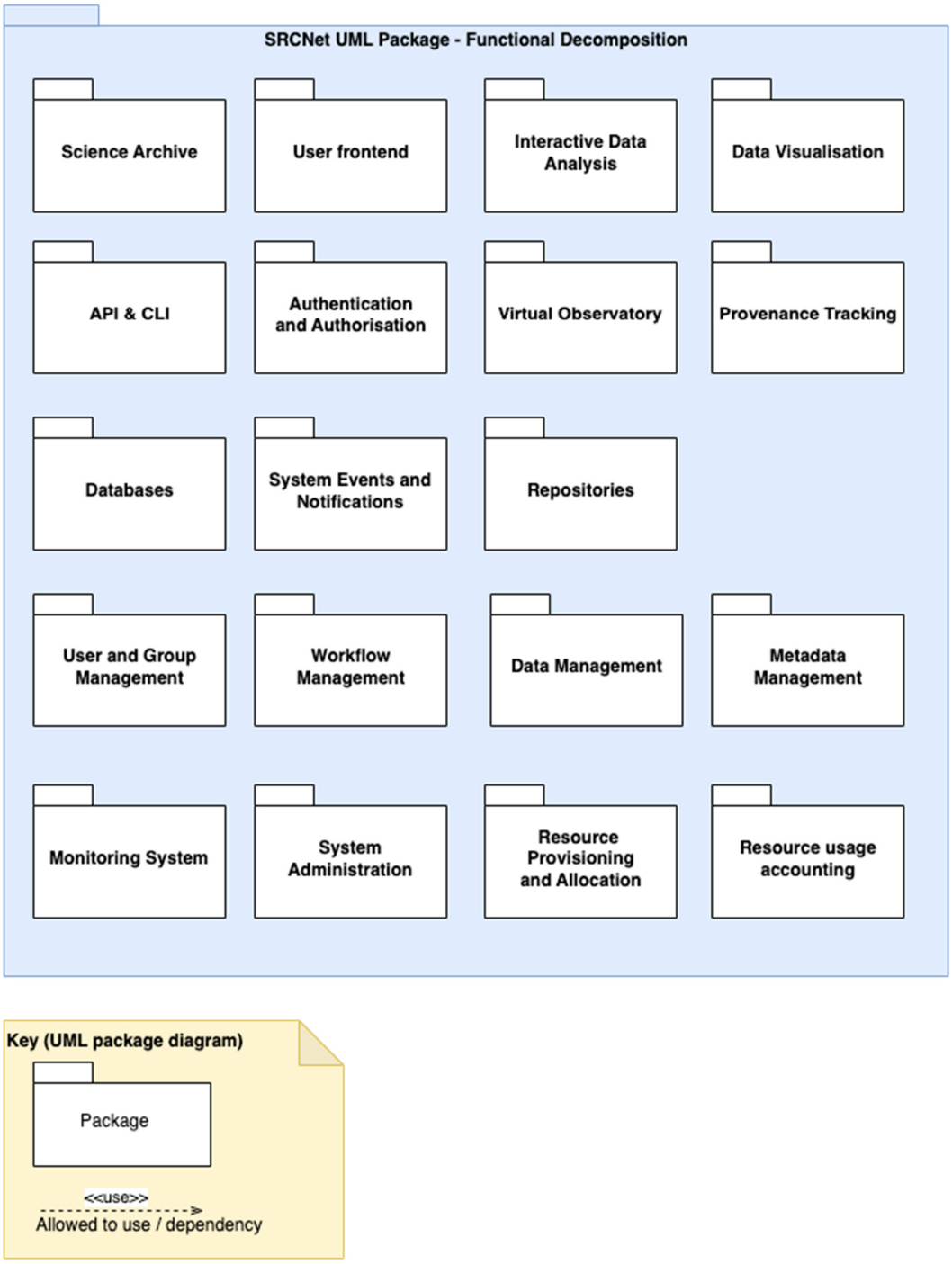

Figure 6. First package decomposition representation of the SRC node blueprint as UML packages.

5.2. Architecturally Significant Design Packages

As can be seen in the functional decomposition diagram, a set of packages have been identified as elements that constitute the SRC blueprint. These packages describe all the different layers of a local SRC instance. These layers have internal connections and, also, connections to other SRCs of the SRCNet.

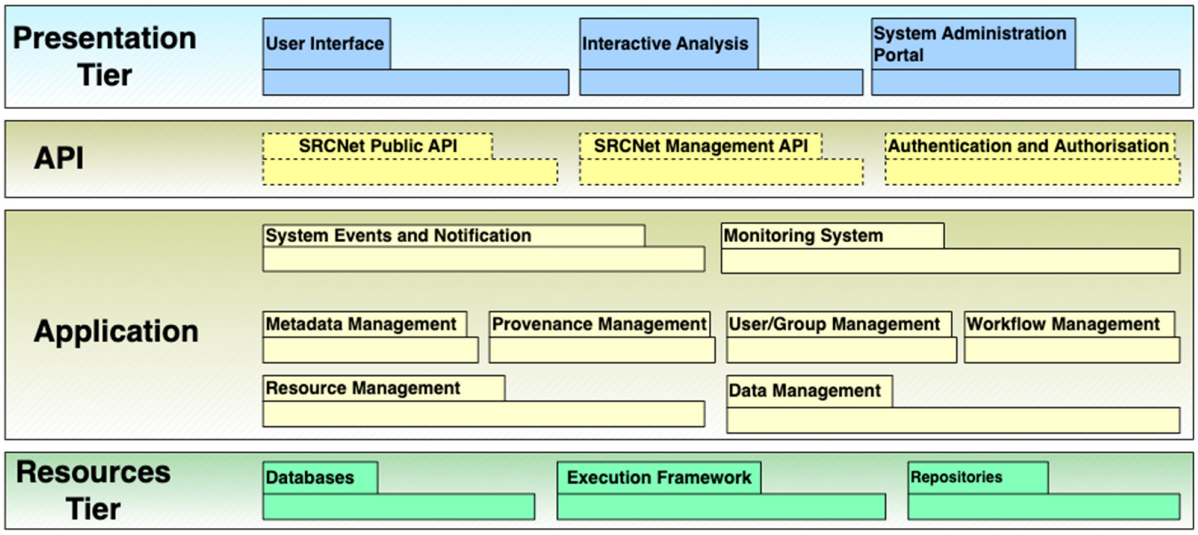

Figure 7. Condensed SRC node architecture footprint, including layers.

5.2.1 Presentation Tier

The presentation layer package represents a parent package composed of the user entry points to the SRCNet platform. This would be composed, among others, of a generic User Interface portal, an interactive analysis system and a system administrator portal. The connection to inner layers is done through a public (exposing the services functionalities) and a management API (used by the System Administration Portal) that could be used by user layer applications or by clients that directly interact with it, bypassing the user interface (e.g. using a command-line client or a Virtual Observatory one).

5.2.1.1 User Interface

Package that represents the User Interface portal to the SRCNet platform. A possible implementation approach would be a Web Portal that would contain, among others:

- An interface to discover, select, analyse and visualise data from the SRCNet.
- An interface to discover capabilities of the SRCNet nodes and the option be redirected to a selected one.
- An interface to create and run workflows.
- An interface to user and group management.
- An interface to explore and share files from user/group areas.
- An interface to explore the documentation and the knowledge base.
- An interface to connect to the support/helpdesk.

5.2.1.2 Interactive Analysis

A client application that allows the execution of this analysis on the SRCNet servers. This client application will use a set of client libraries to interact with the data of the SRCNet (and public data from other projects using access protocols) and, also, other science analysis libraries of common use. A traditional implementation could be a notebooks portal where users can invoke different execution environments and analyse the data resources interactively using a limited set of programming languages or a more advanced future implementation.
Also, a set of specific libraries and methods to facilitate the exploration of the data should be provided and can be considered a sub-package of this one.

5.2.1.3 System Administration Portal

A package that is composed of a set of tools to monitor and administrate the SRCNet. These tools are restricted to be used by the operations team in order to allow them to check the status of the different elements of the SRCNet and solve possible problems. Also, the portal will allow access to shared recovery scripts (if applicable) from the software repository, e.g. disaster recovery scripts to restart failing systems or data recovery scripts to restore backup data. Other recovery scripts could have only a local scope and they should be handled by the support members of the SRCNet Operations team and local IT personnel, locally.
If possible, execution of recovery procedures should be enabled through this portal for remote execution, allowing operators located in different time zones to maintain the SRCNet up and running 24/7 (at least for maintenance tasks that could be executed remotely). The administrator portal will connect to the SRCNet Management system for most of the actions.

5.2.2 Application Tier API

5.2.2.1 Authentication and Authorisation

Package that contains the modules for the federated authentication system and, also, the authorisation rules to access data, invoke services, reserve resources, or access personal/group information and data.
The authentication system should use federated protocols that all the SRCs could use and, also, would allow different identity providers.
The authorisation system should follow the same access rules for all the SRCs and the same security rules.
Exact API methods will be documented in a separate document. These Interface Control Documents will describe API signatures and will contain alignment with local government policies.

5.2.2.2 Public API

Definition of the public server access methods of the SRCNet node and libraries to expose them. Methods will be exposed through a consistent interface (e.g. *RESTful* API). This interface could be used to abstract away the underlying implementation details of an individual node.
This Public API will expose individual SRCNet nodes' functionalities consistently, preventing clients (or User Interface implementations) from requiring particular technical information about the invoked node.

The common Public API should be implemented by all the SRCNet nodes and it could be extended locally for extensions.

Exact API descriptions and methods should be documented in separate documents. As changes to the API impact the compatibility of the nodes, they should be done in a controlled manner, defining backward compatibility for minor versions and possible non-backwards compatible changes for major versions.

5.2.2.3 Management API

Definition of the private server access methods of the SRCNet node and libraries to expose them. These methods, exposed in a similar way to the Public API, would require private access and only would be invoked by the SRCNet Operations team. That could include, among others, security access, monitor access, management actions on the SRCNet nodes, User and Group handling, etc
Exact API methods should be documented in a separate document.

5.2.3 Application Tier

5.2.3.1 System Events and Notifications

Events bus system to handle system events and notifications within the SRCNet nodes and, possibly, between nodes. This event bus approach would allow, e.g. the execution of tasks or communication between modules using a scalable approach.

5.2.3.2 Monitoring System

Package that includes the system to remotely monitor the behaviour of the different SRCNet nodes and the SRCNet as a whole. That would include the invoking methods to query for load, resource allocation, network status, user notifications, energy usage, etc. Also, it would include any needed server module to provide stats homogeneously and, finally, contain an interface to be used by the Operations team to inspect the monitoring result, create reports and take actions on the system.

5.2.3.3 Provenance Management

Package that includes provenance metadata tagging and the subpackage to retrieve the information for the rest of the SRCNet node packages. This provenance metadata would include, e.g. information about the pipeline used during the data creation, including configuration and execution parameters description, data location, data interconnection, etc. This would be reflected in a format that could be consumed by engines in the way of Entities, Activities, and Agents.[8]

5.2.3.4 Data Management

Package composed of the different libraries in charge of the data logistics, including the distribution of data within the SRCNet, access to the data, replication of data during analysis, etc.

[8] Check W3C provenance DM https://www.w3.org/TR/prov-dm/ and IVOA Provenance DM https://www.ivoa.net/documents/ProvenanceDM/ for further understanding of Entities, Activities and Agents definition and interrelation.

5.2.3.5 Metadata Management

Package composed of the different libraries to access metadata from the SRCNet system. This package would be composed of database access methods and query languages to access them. Metadata that could be provided include not only information related to SKA or closer missions like catalogues or observational metadata but, also, other metadata information relevant to the system like users and groups information, saved system events and notifications, etc.
Provenance Management System will connect to this system whenever certain metadata needs to be inserted, updated or queried from a database.

5.2.3.6 User and Group Management

Package responsible for storing, accessing and updating information from the users and groups' profiles. This information could be composed of the users' and groups' details and, also, access rights to the data and other SRCNet resources, the current status of use of the system, quotas, etc.

5.2.3.7 Workflow Management

Package responsible for creating, parsing, storing, and accessing execution workflows compatible with the SRCNet execution framework (5.2.4.2 Execution Framework) which, either accessing local resources or other SRCNet nodes' execution frameworks, will execute them. This module will manage both user-defined workflows and standard workflows provided by SRCNet partners.
Workflow versions will be also maintained aligned and consistent across the SRCNet thanks to this module.

5.2.3.8 Resource Management

Package responsible for handling the allocation of resources of the SRCNet including computing, repositories, update of quotas, etc. Also, resource management would be responsible for the reservation of resources.

5.2.4 Resources Tier

5.2.4.1 Databases

Physical databases that contain the metadata of the SRCNet. That could include among others:

- Provenance metadata associated with products
- Scientific metadata associated with products
- Physical location of products
- User metadata and proposals information
- Catalogues and other scientific metadata ingested in databases
- SRCNet topological information like SRCNet nodes storages, services, computing resources, percentage of usage, node status information, etc.

5.2.4.2 Execution Framework

Module to orchestrate and execute user workflows or any other computing execution needed to analyse the data into the computing resources available within the SRCNet nodes. The execution framework allows the use of multiple, heterogeneous compute resources.

5.2.4.3 (Data) Repositories

Physical data layer associated with the SRCNet nodes that contain SKA pipeline products data (ODPs, ADPs, PLDPs and other possible data products), stored execution entities (software containers), saved workflows and, in user storage areas, intermediate and output data produced by users data analysis, etc.

5.3 Architecturally Significant Design Packages: 2nd level decomposition

A second decomposition of the design package can be done, including sub-packages and tentative interactions, view that will be developed later into the use cases architecture assessment and into the implementation view.

The sub-packages are defined in the next figure

https://confluence.skatelescope.org/display/SRCSC/SRCNet+Modules+View+v0.1

and in section 5.3.1 Sub-Package Descriptions.

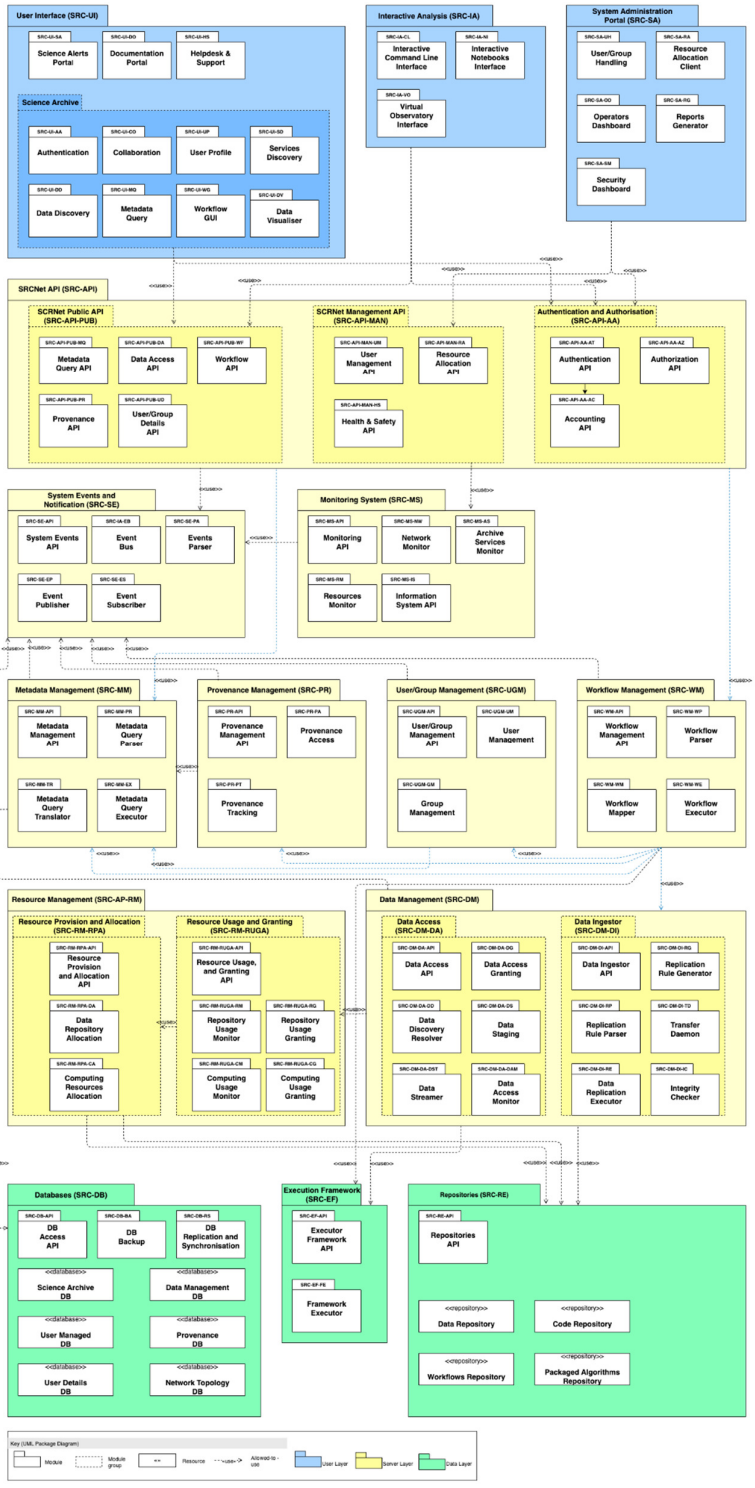

Figure 8. Sub-packages decomposition.

5.3.1 Sub-Package Descriptions

Next table shows the second decomposition level of the components into sub-packages. These packages have different scopes:

- **Local**: Package/service is deployed in the different SRCNet nodes. In most of the cases, an official implementation is provided and a federated deployment approach is followed to guarantee same versions running in all the SRCNet nodes (in particular, for security patches). E.g User Interface to access the Science Platform
- **Global**: Usually a service that is deployed on a limited number of SRCNet nodes. These deployed services are providing support to all the SRCNet nodes, usually by a local/partial implementation that connects to a remote global implementation. Several deployments could be foreseen in the network to provide high availability or load-balancing capabilities. E.g. Metadata Management System to prevent the need for database replicated servers on all the SRCNet nodes
- **Federated**: Pure federated services that are shared by all the SRCNet nodes. All services contribute to these services. E.g. Data Lake Repositories, Federated computing services, etc

#	Package Code	Element	Description	Scope
SRC-AP-UI User Interface				
1	**SRC-UI**	User Frontend	Any functions or issues related to user interfaces, such as GUI, portal(s), user-end application, user interaction, visualisation	**Local** multiple deployments in different SRCNet nodes to provide similar user experience Common functionality, will be shared as **global** modules by all the SRCNet nodes
1.1	**SRC-UI-SA**	Science alerts portal	Portal to receive and visualise astronomical science alerts, mainly in VOEvent format, from alerts brokers	**Local** presentation layer extension only present in some SRCNet nodes
1.2	**SRC-UI-DO**	Documentation Portal	Web location where all the different documentation resources are published to the	**Global** documentation shared by all SRCNet portals

			users like user guides, SKA products definition, examples, etc.	Possible extended documentation for local communities (including muti-language versions)
1.3	**SRC-UI-HS**	Helpdesk and Support	Helpdesk entry point for users to ask questions, report problems and ask for support	Possible multiple entry points but **global** underlying communication channel
1.4	**SRC-UI-AA**	Authentication interface	Interface for federation and identities.	**Global** common service, with possible multiple entry points for load-balancing/high availability
1.5	**SRC-UI-CO**	Collaboration	Interface to allow collaboration actions like sharing data and software, creating groups, managing user areas, etc.	**Local** entry points sharing common client module connecting to **Global** server module
1.6	**SRC-UI-UP**	User Profile	Interface to manage user profile details	**Local** entry points sharing common client module connecting to **Global** server module
1.7	**SRC-UI-SD**	Services Discovery	Interface to discover software, services, data releases and features in different nodes of the SRCNet	**Local** entry points sharing common client module connecting to **Global** server module
1.8	**SRC-UI-DD**	Data Discovery	Interface to discover data from the SRCNet	**Local** entry points sharing common client module connecting to **Global** server module

1.9	**SRC-UI-MQ**	Metadata Query	Interface to discover science entities, query catalogues and perform metadata queries using direct SQL/ADQL languages	**Local** entry points sharing common client module connecting to **Global** server module
1.10	**SRC-UI-WG**	Workflow GUI	Interface to create and handle workflows to be sent to the SRCNet for execution	**Local** entry points sharing common client module
1.11	**SRC-UI-DV**	Data Visualisation	Any functions or issues related to the visualisation	**Local** entry points sharing common client module
2	**SRC-IA**	Interactive Analysis	Any functions or issues related to interactive data analysis, tools, applications, environments, virtual notebooks etc. that bypass the user interface tools	**Local** entry points sharing common client module Possible **local** extensions
2.1	**SRC-IA-CL**	Interactive Command-Line Interface	Library composed by libraries to access data, routines and execution of processes into the SRCNet using the command line. This is usually written in a typical scientific algorithm language (e.g. one or more of Python, Scala, Java, etc.) and they can be executed into an interactive interface (e.g. a notebook) or into a shell script	**Global** libraries

2.2	**SRC-IA-NI**	Interactive Interface	User interface, usually web-based, that allows scientists to write and execute scripts (e.g. a notebooks environment)	**Local** entry points sharing common client module to have similar user experience
2.3	**SRC-IA-VO**	Virtual Observatory Interface	Implementation of IVOA standard interfaces	**Global** services. They can be deployed in different SRCNet nodes
3	**SRC-SA**	System Administrator Portal	Portal, usually web-based, that allows SRCNet administrators to monitor and receive health alerts from the SRCNet. Also, a portal to modify user details, computing and storage quotas per user/groups, permissions, etc.	**Global** portal. It could be deployed in different SRCNet nodes
3.1	**SRC-SA-UH**	User Group Handling	Interface to create and modify users and user groups	**Global** service. It could be deployed in different SRCNet nodes
3.2	**SRC-SA-RA**	Resource Allocation Client	Interface to modify resources allocation per user or group, including quotas and permissions	**Global** service. It could be deployed in different SRCNet nodes
3.3	**SRC-SA-OD**	Operators Dashboard	Interface to show the status of the different components of the SRCNet and specific SRC metrics. Some examples are network status, resources capacity,	**Global** service. It could be deployed in different SRCNet nodes

			both at repository and computational resources point of view, ingestion status, etc.	
3.4	**SRC-SA-RG**	Reports Generator	Module that allows the creation of reports on the status of the network during a certain time scale	**Global** service. It could be deployed in different SRCNet nodes
3.5	**SRC-SA-SM**	Security Dashboard	Interface to show security alerts produced by the SRCNet nodes. That would be also produced by regular security tests on the system on possible vulnerabilities	**Global** service. It could be deployed in different SRCNet nodes
4	**SRC-API**	SRCNet API	API to publish all the possible access methods to the SRCNet node. This API should be consistent between the different nodes and could be extended by particular features implemented in specific SRCs SRC-AP-APIs are used by the presentation layer but they should be designed in a way that could be also accessed by scripting interfaces (bypassing the presentation layer) or by other	Common definition with **local** deployment

			clients so they would make use of standards or they should be properly documented.	
4.1	**SRC-API-PUB**	SCRNet Public API	API methods used directly or indirectly by users, either by using tools or the scripting interface	Common implementation with **local** deployment
4.2	**SRC-API-MAN**	SCRNet Management API	Protected API methods used directly or indirectly by administrators	Common implementation with **local** deployment
4.3	**SRC-API-AA**	Authentication and Authorisation	API methods to be used for Authentication, Authorisation and Accounting	**Federated** service
5	**SRC-SE**	System Events and Notifications	Any issues related to the event-driven functionality, event bus, signals, notifications, events, event channels, produces-subscriber model, event brokers, etc.	**Local** module including **federated** services
5.1	**SRC-SE-API**	System Events Interface	Interface to subscribe and publish events and notifications to the SRCNet	**Local** module connected to federated event bus
5.2	**SRC-SE-EB**	Event Bus	Federated Event Bus used by the SRCNet	**Federated** service

5.3	**SRC-SE-PA**	Events Parser	Shared library to parse and create Events that can be imported by Subscribers and Publishers	**Local** module using common library
5.4	**SRC-SE-EP**	Events Publisher	Module to handle event publishers	**Local** module using common library
5.5	**SRC-SE-ES**	Events Subscriber	Module to handle event subscribers	**Local** module using common library
6	**SRC-MS**	Monitoring System	Module that handles the monitoring of the elements of a particular SRC like e.g. network, resources and services. Send notifications and events to the Event bus to be consumed by the monitoring dashboards	**Local** module using **federated** service. Communication between nodes of the monitoring metrics could be implemented using centralised or hierarchical models
7	**SRC-MS-IS**	Information System	The monitoring system will update the status of the computing resources, occupancy of the different repositories, network connectivity between nodes and other metrics that, in combination with the static information from the network, would produce a topological view of	SRCNet nodes could have a **local** client or a **global** service. Global service could be deployed in different SRCNet nodes and synchronised between nodes

			the SRCNet in a particular moment. This information will be stored in the Network Topology Database and it would be queryable through the information system, either by the system administration portal, the execution planner or any other system	
7	**SRC-MM**	Metadata Management	Middleware to parse, send for execution and stream results of metadata queries	SRCNet nodes could have a **local** client using a common module or a **global** service. Global service could be deployed in different SRCNet nodes and synchronised between nodes
7.1	**SRC-MM-API**	Metadata Management API	API access methods to send queries to the system and retrieve results, to be used by the rest of the modules and preventing direct access to the database	**Local** module using common implementation
7.2	**SRC-MM-PR**	Metadata Query Parser	Module to parse received queries, analyse security of the queries and check consistency	**Local** module using common implementation
7.3	**SRC-MM-TR**	Metadata Query Translator	Module to translate queries received in a certain query language (e.g. in	**Local** module using common implementation

			ADQL, SQL,..) to an executable format	
7.4	**SRC-MM-EX**	Metadata Query Executor	Module responsible to send a query for execution to the relevant data source (usually a database) and usually using a pooling mechanism	**Local** module using common implementation
8	**SRC-PR**	Provenance Management	Subsystem to track and access provenance information from digital objects	SRCNet nodes could have a **local** client using a common module or a **global** service. Global service could be deployed in different SRCNet nodes and synchronised between nodes
8.1	**SRC-PR-API**	Provenance Management API	Interfaces published to access the provenance management functionality	**Local** module using common implementation
8.2	**SRC-PR-PA**	Provenance Management Access	Module to handle the provision of provenance metadata to other modules	**Local** module using common implementation
8.3	**SRC-PR-PT**	Provenance Tracking	Module to create and update provenance metadata for digital objects	**Local** module using common implementation
9	**SRC-UGM**	User and Group Management	Any functions or issues related to	SRCNet nodes could have a **local** client using a common module or a

			managing users and groups	**global** service. Global service could be deployed in different SRCNet nodes and synchronised between nodes
9.1	**SRC-UGM-API**	User and Group Management API	Interfaces published to access the user and group management functionality	**Local** module using common implementation. It connects to a local or global module
9.2	**SRC-UGM-UM**	User Management	Module to manage user information	**Global** module. Global service could be deployed in different SRCNet nodes
9.3	**SRC-UGM-GM**	Group Management	Module to manage user groups information	**Global** module. Global service could be deployed in different SRCNet nodes
10	**SRC-WM**	Workflow Management	Subsystem to handle the creation, analysis and execution of scientific and system workflows	SRCNet nodes could have a **local** client using a common module or a **global** service. Global service could be deployed in different SRCNet nodes and synchronised between nodes
10.1	**SRC-WM-API**	Workflow Management API	Interfaces published to access the workflow management functionality	**Local** module using common implementation. It connects to a local or global module
10.2	**SRC-WM-WP**	Workflow Parser	Module to handle workflows to check consistency and security aspects	**Global** module. Global service could be deployed in different SRCNet nodes

10.3	**SRC-WM-WM**	Workflow Mapper	Module to prepare workflows for execution, adapting them to the SRCNet resources	**Global** module. Global service could be deployed in different SRCNet nodes
10.4	**SRC-WM-EX**	Workflow Executor	Module responsible for sending the workflow for execution to the resource layer, usually using a pooling mechanism	**Global** module. Global service could be deployed in different SRCNet nodes
11	**SRC-RM**	Resource Management	Any functions or issues related to the accounting of resource usage and quotes	SRCNet nodes could have a **local** client using a common module or a **global** service **Global** service could be deployed in different SRCNet nodes and synchronised between nodes
11.1	**SRC-RM-RPM**	Resource Provision and Allocation	Module responsible to identify required resources for a certain task, e..g. workflow execution, into the SRCNet and allocating them to other modules	**Global** module. Global service could be deployed in different SRCNet nodes
11.2	**SRC-RM-RUGA**	Resource Usage and Granting	Module to identify a resource allocation request and compare with user/group quotas, the status of the SRCNet resources, security rules and execution priorities	**Global** module. Global service could be deployed in different SRCNet nodes

12	**SRC-DM**	Data Management	Any functions or issues related to managing data, global data transport and management, local data movements, data backups and replications	SRCNet nodes could have a **local** module with the data access request capabilities or a **global** service including the ingestor.
12.1	**SRC-DM-DA**	Data Access	Module responsible to access digital objects present into the SRCNet	**Local** module connecting to **global** service
12.2	**SRC-DM-DI**	Data Ingestor	Module responsible to ingest digital objects into the SRCNet and create requested copies across the network	**Global** service. It could be deployed in different SRCNet nodes for high availability
13	**SRC-DB**	Databases	Resource layer for metadata repositories, usually in the form of databases	**Global** services. It could be deployed in different SRCNet nodes for high availability but they will require synchronisation
14	**SRC-RE**	Repositories	Resource layer for data repositories, usually in the form of digital objects. This could be data files, source code, workflows and packaged algorithms (e.g. in the form of containers)	**Federated** service
15	**SRC-EF**	Execution Framework	Module to orchestrate and execute user workflows or any	**Federated** service to abstract compute resources. Different implementations could

			other computing execution needed to analyse the data into the computing resources available within the SRCNet nodes. The execution framework allows the use of multiple, heterogeneous compute resources. It is instantiated by the workflow management system	be provided and shared by the SRCNet for different technology flavours

5.3.2 Services View

This section sketches some of the services that will be provided by the different SRCNet nodes. This list should not be considered exhaustive as new services could be defined and agreed upon. Exact APIs for these and future services must be defined into agreed ICDs that must contain a proper granularity level in terms of publication protocols, input parameters, output results, serialisation of objects and flow and it should allow APIs signatures evolution by modification of the existing methods or definition of new ones. However, it has been considered important to illustrate how the modules are converted into services for a complete understanding of the architecture.

5.3.2.1 Data Management Service

This service is mainly provided by the deployment of a Data Management system. As mentioned in the sub-packages table, it is composed of two main services:

- Data Access (**SRC-AP-DM-DA**): The service in charge of accessing the data, invoking data replications into staging areas, streaming data from the RSEs, etc. Also, this service could be used too by data parser libraries (e.g. data mesh server parser libraries as described in Section) to access data
- Data Ingestor (**SRC-AP-DM-DI**): The service in charge of the data dissemination within the SRCNet for data produced by the telescopes and, also, to disseminate data between SRCs. This service maintains up-to-date metadata information of data locations and ensures that the data is properly transferred (e.g. by verifying checksums) and that the repositories' content is consistent (e.g by identifying orphans files or missing links to the data)

The data management service can have different levels of incarnations. Not all the SRCNet nodes should implement the data ingestor part (responsible for handling the data dissemination) but all the data management services should have a data access service (and connect to the data ingestor instances whenever needed to invoke data replications).

Connection to this service through the SRCNet API will be done by the User Interface and, also, by the use of command line clients written in common use languages (e.g. python, Java, etc) developed and maintained by SRCNet development teams.

5.3.2.2 Metadata Management Service

This service is mainly provided by the deployment of a Metadata Management system and allows it to query content from databases located in the SRCNet and, also, to allow the upload and data tables creation (using roles control).

For the query side, the following expected queries are expected:

- Discovery of data within the SRCNet data lake by a query for data locations using data identifiers.
- Discovery of science data entities by using scientific data parameters (e.g regions of the sky, instrumental configurations, etc.).
- Queries on scientific data present in the database (e.g. catalogues).
- Discovery of user and group details and associated access permissions.
- Discovery of SRCNet details (e.g. services running, monitoring alerts and notifications, network topological aspects, etc).

For the data upload, some main uses are foreseen:

- Allow users/groups to upload tables into their personal areas, including control on the sharing level.
- Allow the data ingestor system to insert and update the data locations database into the respective database.
- Allow the User Management system to modify content on the User and Groups database.

All SRCNet nodes should implement the metadata query system with different levels of incarnations. A global implementation should have databases locally accessible and these metadata management systems should allow query executions, table uploads and table content updates. Others, classified as local, will just redirect the queries to the *global* deployed instances. In both cases, the interfaces for the clients invoking these services should be indistinguishable.

5.3.2.3 Workflow Management Service

As expressed in different parts of this document, the SRCNet will implement different levels of federated computing approaches. One of them, based on a computing execution service broker, could be used to abstract the complexities of execution of workflows into a heterogeneous computing resources network like the ones foreseen for the SRCNet.

This service could be implemented following an interface as follows:

- Workflow serialised document in an agreed format (e.g. Common Workflow Language (CWL) or similar) is sent to the interface. This document must contain exact descriptions of input parameters, software elements used (including versions), hardware execution dependencies and configurations, etc.
- The Workflow Management system will decompose and analyse the workflow and the Execution Planner would be used to identify an acceptable execution plan using characteristics of the information of the SRCNet topology extracted from the Information System, location of the data, hardware dependencies, etc. A more detailed description is present in 6.7. Computational resources allocation and federated execution

- After decomposition, the Resource Management systems will interoperate and invoke the executions to their Execution Framework local components.

All SRCNet services should have a workflow management service to coordinate the federated execution.

5.3.2.4 Authentication and Authorisation Service

All SRCNet nodes must have an authentication service entry, either a global service that connects to federated identity providers or local services implementation connecting to the global SRCNet authentication declared services.
All SRCNet nodes must integrate consistent authorisation modules to ensure a secure access system following, e.g. the SKA data access policies.

5.3.2.5 Services Discovery Service

All SRCNet nodes must declare their running services (common and extensions) using a consistent service discovery service, providing all the needed metadata to invoke them (e.g. ports, service URLs, service signature, etc), number of running instances, resources allocated, etc. The information provided by the SRCNet nodes could be dynamic as new services could be started.

This metadata will be compiled into the information system tables to be used by other components (e.g. by the execution planner).

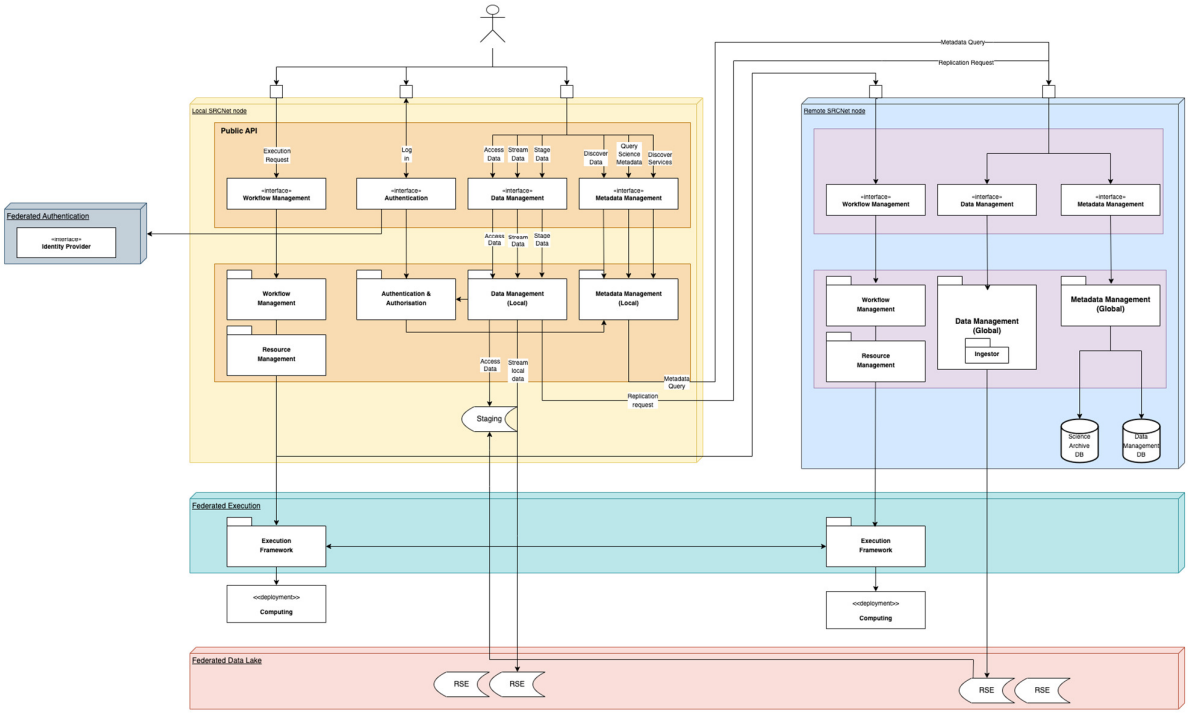

Figure 9. Services View

View of some of the services implemented by the SRCNet nodes. Four different sets of interfaces are shown: Authentication interface, Workflow Management interface (to execute workflows and other execution entities into the node), Data Management interface (to access data and request replications), Metadata Management interface (to query for data locations, science entities or network services). Data and Metadata Management systems could be fully implemented (global) or just as a connector to a global service. Federated Execution could be done through connections between the execution frameworks or through an interface. Federated Data Lake is composed of Resource Storage Elements available and published to the Data Management System. Shown interfaces could be expanded with other methods.

https://confluence.skatelescope.org/display/SRCSC/Global+vs+Local

6. Process View

In this section, we sketch some relevant process views of different parts of the SRCNet system. These descriptions will be as agnostic as possible on the implementation and definition of the software and technology stacks. Also, some of the processes described could require a more detailed definition in a dedicated Interface Control Document (ICD) or a detailed implementation document.

6.1. Ingestion of data into SRCNet

Science data produced by the two SKA telescopes locations would need to be distributed into the SRCNet at a comparable rate of the data production. The main reason is, not only, to allow the SKA data analysis

by the scientific community after production as soon as possible but, also, to prevent the blocking of the operations of the telescopes because the data dissemination interface to the SRCNet is the operational bottleneck. Both locations will have a storage system with the scientific data result produced by processing the telescope's raw data. Although this processing will reduce drastically the amount of data to be distributed into the SRCNet, these local storage systems should be only considered a temporary data staging area. If these areas are full at a certain moment, this would impact real-time operations.

Following an agreed procedure, the SRCNet data management system will have access to notifications of data available for distribution. Once a new data available message is received, the data management system will identify the final location within the SRCNet or “primary copy”. This location would be one of the high-read access storage systems or hot tiers (the requirements on the read access rates of the primary copies could depend on the data type and other factors) provided by the SRCs members of the SRCNet. To do this distribution, some points, among others, should be considered, for example:

- Resources available for the different SRCs (repositories and computational resources)
- Network connectivity
- Location of programme proposal members that requested the observation (if any)

At least two copies of the data are expected to be distributed to the SRCNet. The “secondary copy” of the data could be replicated within the SRCNet to prevent data loss or loss of data availability due to, at least, a problem in one particular SRC (these copies could be located in a not in the optimal read access storage systems but with enough performance to support online access) or could be generated in hot tiers to ensure low latency access to popular datasets. Popular datasets will be identified through the monitoring and accounting of the data access operations. The number of “secondary copies” would depend on the resources available.

The heuristic algorithm to decide the SRCNet node to be used as the destination of the primary data distribution would be adjusted during operations through close monitoring of the system resources and access response, taking into account all the possible elements.

The SRCNet should have different levels of repositories, from hot storage for high-performance analysis to cold storage for old data or data not used so often during analysis. This is why the SRCNet data lake storage should follow a Hierarchical Storage Management (HSM) approach, which different storage tier layers with different performance. Currently, the proposed strategy would be to maintain one year of the science data products in hot storage (e.g. SSD), the Online Tier, and a cooler technology for older data (the SDP cost model has assumed a costing based on SATA Hard Disk Drives). However, the selected model would be discussed during the implementation phase, in a balance between performant science analysis capabilities and SRCNet partners' budgets.

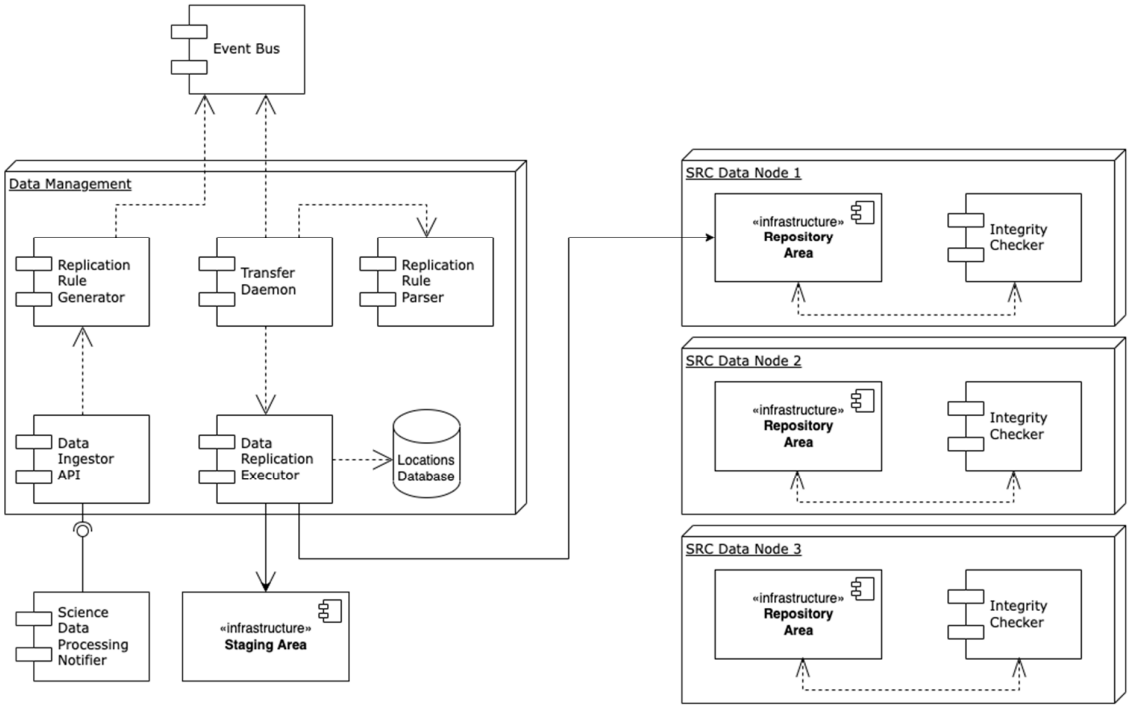

Figure 10. Data ingestion from telescopes locations into the SRCNet.

Science Data Processing modules alert of new data available in a staging area using a declared API. The Data Management module will invoke a replication rule to request a data movement to a certain SRC location. A pooling mechanism like an event bus implementation (TBD) is used. A daemon is reading the pool and raising data replication execution from the staging area to one or more repository areas within the SRCNet. After transfer, an integrity check is done on the transferred data and a data locations database is updated reflecting the existence of a new data copy.

https://confluence.skatelescope.org/display/SRCSC/Data+Management+Ingest

6.2. Data access during processing SRCNet

There are two basic ways for the modules to access SKA data inside the SRCNet repositories:

- Access to data that are at the same location where the module is located (local SRC)
- Access to data that are in a remote SRC location

For the first case, a direct stream to the digital object representing the data could be obtained. Also, there exists the possibility to mount directly the storage elements to be visible for the modules as file storage for some types of data.

However, remote data access is more complicated. One way to convert remote data access analysis to local data analysis is to invoke the creation of a local copy of the data by producing a data replication rule. This staging data approach is, in the case of very big objects, not optimal because it introduces latency in the data transfer but it could be used if the data is going to be accessed during a long period of analysis (preparation of environment) or if the data to be used are not large. This approach is described in 6.2.1. Data staging approach.

Another way to access remote data for analysis is by the creation of modules that can read and process remote data copies. Parser modules could perform remote operations on the data and only transfer back the result of those operations, therefore avoiding potentially costly data movement. This is the “data mesh” approach that is described in 6.2.2. Data mesh approach.

Data element size, network connection, the kind of operations on the remote data and other aspects could imply that the “staging approach” or the “data mesh approach” is more optimal, implying that both approaches need to be implemented and only the analysis of the use case will allow deciding which approach is more appropriate case by case.

6.2.1. Data staging approach

Once data access is invoked using the SRCNet API, the data management system should, in this order:

- Check data access rights on the relevant digital object
- Discover where this digital object is located in the SRCNet. Usually, several copies could be present for the same digital object.
- In case a copy of the digital object is located in the local SRC storage close to the computing, data could be directly streamed to the API.
- A request will be made to the Resource Usage and Granting module to check the user/group's quotas before making additional copies of the data.
- In case no copy is present in the local SRC storage, a local copy is requested to the Data Management Ingestor module. This module will raise a replica creation in the target repository. The replica could have an expiration time stamp so this could be cleaned after a certain time. The idea is not to saturate the system with these staging copies. A possible reset of the expiration date could be done if the data is accessed again.
- Notice that this copy will have a registration into the locations database so another request for the same digital object for this SRC will not produce another data replication during the lifetime of this staged local copy.
- Once a local copy is generated, the digital copy could be streamed to the API layer.
- All the processes will be monitored and metrics will be produced for the module to create stats and, e.g. identify popular datasets that could require more copies in the SRCNet.

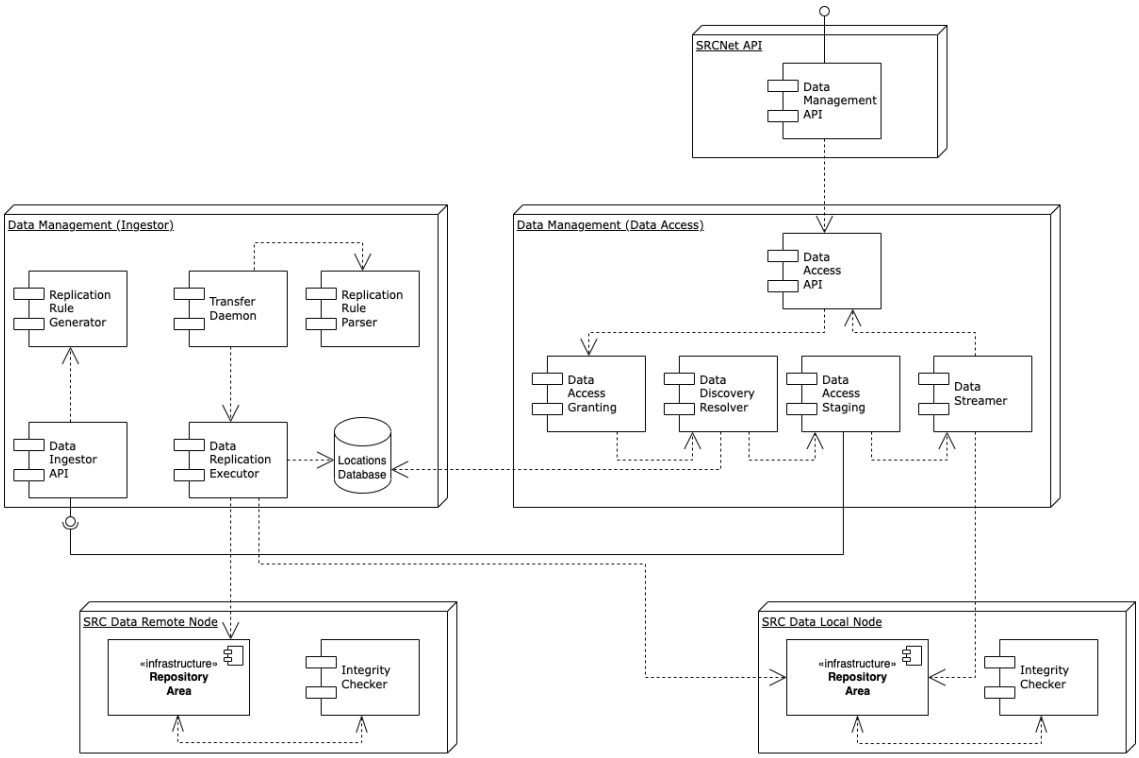

Figure 11. Data access following the "create local copy" approach.

Local copies can be created by invoking the ingestor module if needed.

https://confluence.skatelescope.org/display/SRCSC/Data+Management+Access

Data ingestion and data staging are, in practice, quite similar in the transfer process with the main difference being the persistence of the data into this storage system and the storage system type. During data ingestion, the data is flagged as persistent and, in the data staging, data will be flagged as volatile with an expiration flag that could be adjusted depending on the processing and the computing resources availability.

All the users should also have dedicated, persistent user storage areas defined by a quota, where they should be able to upload, store intermediate and final results of processing and share these files with other users and groups of users. The description of this part will be studied in 6.6. Collaborative environment.

6.2.2. Data mesh approach

A data streamer could be invoked remotely to perform operations on remote data to prevent data movement latency. This is usually described as the data mesh paradigm (Dehghani 2022) where data and parsing code (including hardware dependencies to execute this code) are considered domain data. This paradigm is being implemented by many existing data lake services, many of them in the commercial world, using their implementation flavours.

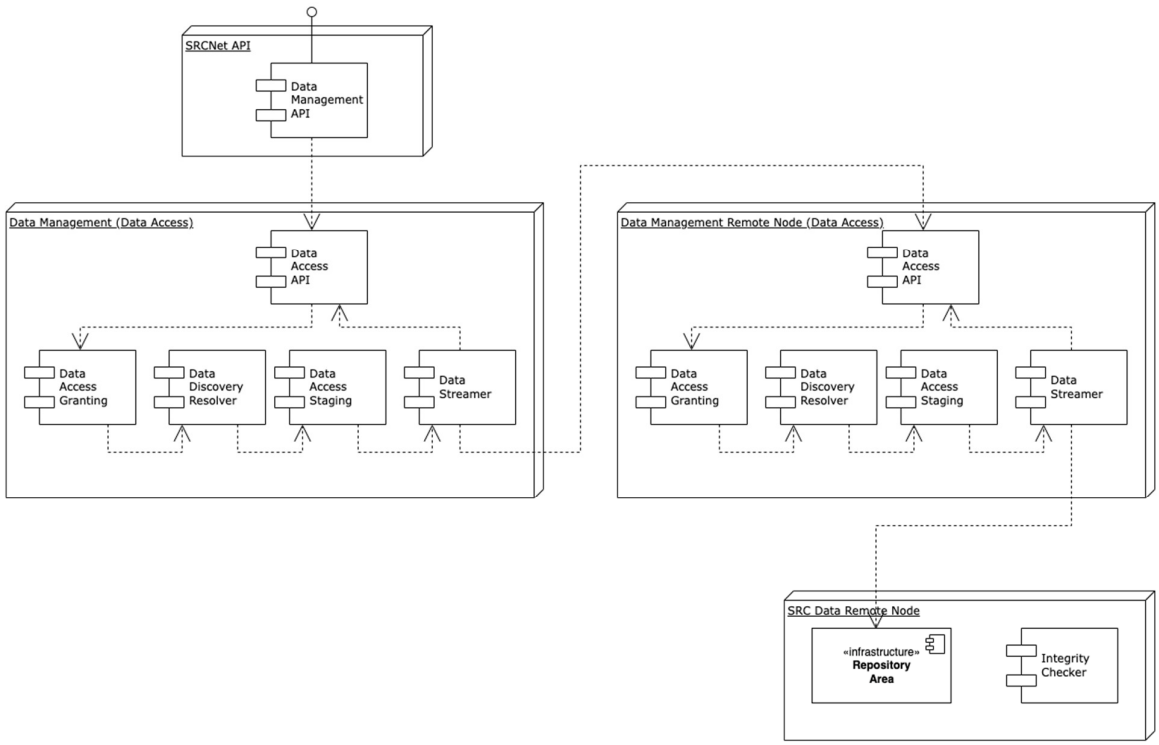

Figure 12. Data access using the data mesh paradigm.

The Local Data Streamer module connects to a remote Data Streamer located in another SRCNet node. This connection invokes a remote operation on the data preventing the latency of the movement of the data so only the result of the operation is transferred from the remote SRCNet node to the local SRCNet node. This could only be done if the operation is included inside a parsing library runnable on the remote data streamer. In both SRCNet nodes, access granting and discovery operations are implemented to prevent security breaches.
https://confluence.skatelescope.org/display/SRCSC/Data+Mesh

To properly implement this paradigm, the data products dependencies (parser libraries and hardware), should be considered on the data dissemination decisions so data should be transferred to nodes where the computational resources are sufficient to perform these parsing operations.

The data mesh paradigm implementation implies a better integration of the computational resources and the data lake. Also, the data mesh paradigm creates a direct integration of the data and the related libraries optimised to perform data operations. These two integration levels are guaranteed by the creation of data domain experts teams (e.g. experts on a particular data type, a particular data exploitation use case or a particular science domain) who are in charge of the development and maintenance of the parsing libraries and specific domain use cases and the hardware dependencies required to execute them. These teams are composed of multi-layer experts, including e.g. data scientists, data engineers and hardware engineers.

Domain data expert teams are defined to ensure different levels of decentralisation and autonomy:

- Decentralisation of business into domains
- Decentralisation of engineering layers approach into self-autonomous teams
- Decentralisation of monolithic into microservices
- Decentralisation of operations into DevOps teams

Using the libraries produced by the domain data expert teams, other teams can integrate and invoke the different methods of the API (locally or remotely) without the need of deep knowledge of the logic of the digital objects structures that are out of their expertise. These libraries can be evolved by the creation of new methods that are published whenever they are ready for use.

In the case of SKA data, this microservices autonomous approach is particularly important for data products with complex data parsing challenges (e.g big data cubes) transforming the operations of parsing and data operations (what could imply data movement latencies and poor performance methods) into an invocation of services created by the domain data experts that are optimised in performance and resources and executed on top of optimal hardware infrastructures.

To implement this paradigm, the data discovery resolver module should also contain a parser service resolver module, indicating not only the location of the data but, also, the existence of a remote parser library ready to be invoked close to the remote data product.

The data streamer module could require, for higher analysis performance, a staging area where the data could be stored temporarily during certain analysis tasks. For efficiency, SRCNet nodes should have, at least, a storage repository where the data could be moved for high-performance I/O throughput, either during the data dissemination or during the preparation of the analysis environment.

6.2.3. Data staging vs Data mesh

Data staging and data mesh approaches could be used on different use cases or work in coordination.

In the case of big data files (e.g. big data cubes), that could produce a high latency when they need to be moved to a different SRC, a data mesh approach could be more appropriate. For this case, when the analysis is done by an execution workflow that could be decomposed and the code distributed to the SRC where the data is, this approach would prevent unnecessary data movements. This approach could be done, for example, for visualisation tools where the data parsing server could be moved or located into all the SRCNet nodes or for operations that could be included in streaming filters close to the origin of the data.

However, data staging could be used when the libraries that access the data do not implement a particular data access method required for the analysis or for operations that require the combination of data from multiple SRCs. For this kind of analysis workflow, a preliminary data environment preparation step is instantiated before executing the data analysis task, moving the required input data to a temporary area, so the analysis could start with all the data already in the analysis node.

6.3. Science Platform User Interface pattern

The architectural representation separates different layers for every SRC node instance. On the top layer, a user interface/command line interface is defined that connects to the Server Layer using a common (for all the SRC nodes) server layer interface. Each SRC Node has minimum functional requirements on the user interface that could be extended with extra functionality depending on the regional/national requirements or capabilities.

Federated Execution is a crucial requirement in the design of the SRCNet as scientific workflows may need to be executed at remote SRCs, i.e. not local to the user, where data are located or where the requisite resources are available. Therefore, it is crucial that the systems are connected, consistent and coherent. That implies that some of the minimum functional requirements to be implemented on the client side should be present in all the SRC nodes so the user experience is similar and, also, to guarantee the interoperability between nodes that could produce the concept of a global SRCNet.

Also, client components should be portable (e.g. in the form of software containers) and reusable by other SRCs to minimise the learning curve of the community accessing the network (similar essential services), provide a coherent view of the system and allow features like federated execution. This also makes the maintenance of the codebase viable and not dependent on individuals.
Essential services should have the same behaviour for all the different SRCNet nodes. Technically speaking, a geographical resolve technique (e.g. GeoDNS) could be used to redirect users to the closest local SRC node so the aspect and functionality should be consistent (although users should be able to manually select a concrete SRCNet node). This also implies that the deployment of the services should be done, in many cases, on different SRCNet nodes to guarantee e.g., reduce latency and increase access performance.

That means that, apart from some obvious functionalities like data discovery, visualisation or the analysis interface, the SRCNet node architecture should be able to accommodate new functionalities (extensible) and execute these services with the proper scalability to support not only the local community but also users from other members of the international community (scalability). This international agreement would ensure that local community members are also able to execute scientific workflows also in remote locations, not limiting the science to be produced. More details of the accounting framework can be found in 5.2.5.2 Execution Framework.

The best candidate for the architecture Presentation tier for the Science Platform Interface is the so-called Gateway Pattern. In this case, every node declares the services that it is going to expose. This configuration of services produces three effects:

- User Layer components of declared features are made available in the User Interface. That could be seen as menu options, clickable icons or any other UX technique that dynamically reflects the configuration of the node.
- Required component API methods for all declared services are dynamically exposed for this SRC node.
- Server components to manage API or any other required method are started on the server side.

Usually, to guarantee scalability, these server components are managed by an orchestrator that starts all the required server components with the required resources of the user, rescaling dynamically the number of processes depending on the use at a particular moment.

The nodes of the SRCNet will share some of the components of the presentation layer (and the underlying server counterparts). This decision is done to minimise the learning curve of the scientists accessing different nodes of the SRCNet and to facilitate homogeneous access to the users (see Figure 9: Gateway pattern to be used on the client side…). Some of the services and components should be common to all the nodes and the orchestration of new versions will be done using what is called the distributed deployment model described below. These common services could include, e.g. a data discovery client, a metadata query system, a services discovery client, etc. That implies that when a new version of a particular module is produced, there will be a way to distribute this new version to the operations team into all the SRCNet nodes.

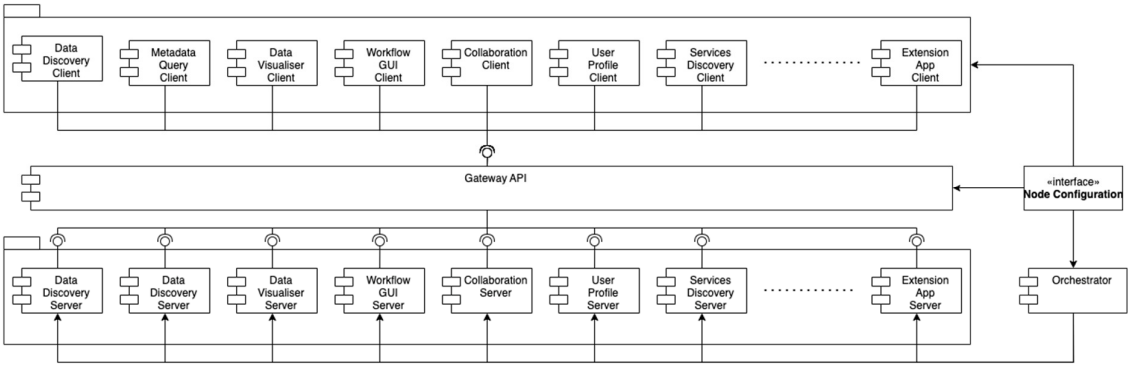

Figure 13. Gateway pattern to be used on the client side.

Functionality elements are declared on a configuration element. The corresponding client elements are then produced on the client side and an orchestrator instantiates and scales the required server elements on the server side to match user demand. The Gateway API reflects the API methods called by the user elements to communicate with the server elements.

https://confluence.skatelescope.org/display/SRCSC/Gateway+Pattern

In this way, the SRCNet will contain different nodes sharing some common basic functionalities and others that could be extensions and SRCNet node-specific. The selection of the node to be used by a user connecting the SRCNet could be based on different criteria like the geolocation of the user (redirecting the scientist to the closer node), the belonging of a particular program in which the data is located or other computational resource considerations. The possibility to force the connection to a particular node by the user should be maintained in case, e.g. the user wants to use a particular functionality (extension) only present in one particular SRCNet node or a particular infrastructure item (e.g. support for GPU accelerations available for a workflow).

This selection of the node of the SRCNet is mainly applicable to the presentational layer (portal node) as during execution a more sophisticated federation execution approach would be developed in phases to execute workflows in remote SRCNet nodes under the hood.

6.4. Metadata Management System

Some of the SRC nodes will have a dedicated Metadata Management System. This module requires to have, in many cases, big databases and dedicated hardware so it is not foreseen the need to have these

resources in all the SRC nodes of the SRCNet. However, the metadata service should be available for all the SRC nodes as this is a basic service so the service will be offered in the client layers of all the SRCs, delegating the execution to the SRC nodes or locations with this capability at the server side.

Queries will be of different types:

- Discovery and query of science entities from user e.g. observational data, catalogues or VO services
- Discovery of digital objects (e.g. data elements)
- Query on provenance metadata
- Queries on user-managed databases
- Queries on user details and group information from different modules of the system
- Queries on system datasets by modules of the system

To reduce the software elements to be implemented, it is intended to cover most or even all of the types of queries using a unique metadata management system, so parsing, translation and execution submodules should be designed to be reusable and database agnostic. Different access to the databases other than the metadata management system could be proposed if major security or access limitations are declared.

Once received a query from the User Layer or command line interface, the query is redirected to the Metadata Management system API that:

- **Parse Query:** The metadata management system will be exposed in the SRCNet node API so traditional User Interface modules and command line interfaces can instantiate it. A supported query language is foreseen (e.g. ADQL, SQL or NoSQL parametrised languages) with possible other supported query languages as extensions but the system should be protected against non-secure access (e.g. SQL injection) by the implementation of a parser module that limits the action of external users on the resources and prevents certain types of unsecured queries
- **Translate Query:** The query should be converted from an externally declared query language to a native query language that the databases can understand. This translation would depend on the final database type used by the SRC and the extensions present in the target database. For example, it is typical to have a geometrical extension like pgSphere, Q3C or Postgis, so the translator could decide to translate a particular ADQL function to the relevant function of one extension that implements it with optimal performance.
- **Security and allocation:** Access rights of the resources involved in a particular query should be studied. For example, it should be ensured that the user performing the query has access to all the tables joined in the query. Also, certain queries (like non-volatile table creation into the user-managed database) require that the resources requested are available for this user (e.g. using a quote system). This is why during the process the Resource Management system (both for resource provision and granting) will be invoked. As the query is usually decomposed inside the translator submodule, this could be a good place to integrate these checks, although a different submodule could be defined if the complexity of the translator increases too much. In case of errors during the

translation, user access or resource allocation are found, an error message will be streamed to the user.

- **Query execution:** Finally, the query is sent for execution. Usually, the query executor maintains a connection pool and dynamic creation of new connections under demand to protect the databases from high peaks. In the case of small databases, an orchestrator could also handle the number of active databases, adjusting the resources to the demand.

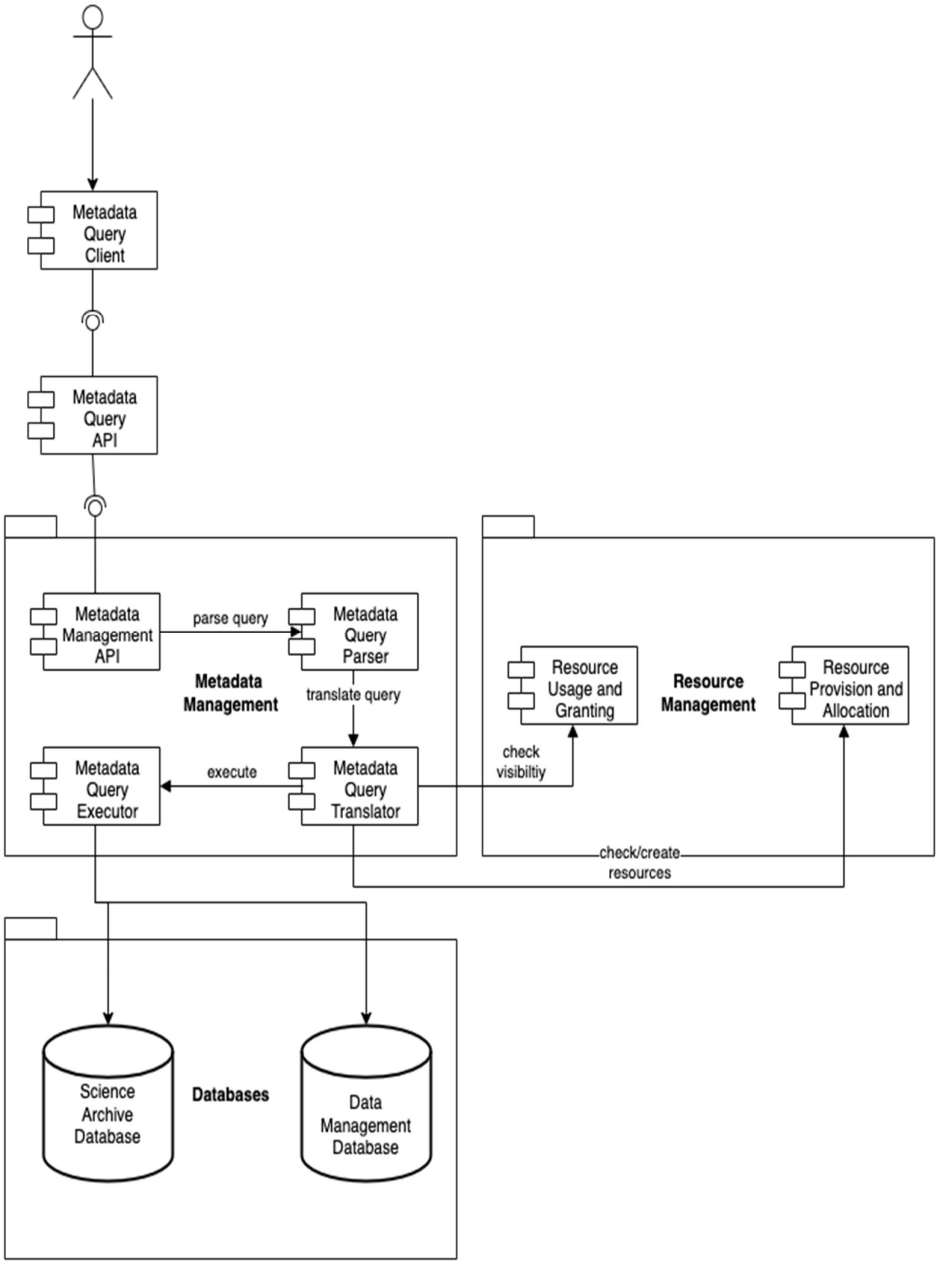

Figure 14. Metadata management system architectural diagram.

Metadata management client functionalities are exposed for the SRC nodes. These functionalities will be connected to a local or remote API where a certain metadata management system will take care of the parsing, translation, check of access and resources and execution of the queries.
https://confluence.skatelescope.org/display/SRCSC/Metadata+Management

6.5.1 Metadata Replication

Usually, the hardware resources needed for a metadata system are specific. Big databases could require high-memory servers for single-node DBs and specific server nodes for parallel databases. Also, the complexity of the configuration of these systems means that the maintenance and upgrade of these servers are usually done by database administrator experts.

For example, for archives like Gaia, the initial configuration to cope with the Gaia catalogue implied a minimum of two servers (for high availability) with 4x12 cores and around 1.5 TB of RAM.

The consequence of the need for this exotic hardware is that it would not be compulsory to have a complete metadata module deployed in all the SRCs but just ensure that the module redirects queries to certain registered SRC full-fledged metadata systems. However, to have high availability using the geographical distribution, a minimum of two full-flesh services should be deployed.

Alternatives are

- **High availability:** One primary server provides the service at one SRC. This SRC should have, internally, local load-balancing or high-availability support. A secondary service in a different geographical zone is prepared and maintained aligned with the primary server through stream replication. The secondary server could be converted as primary in case of problems on the primary SRC metadata management system.
- **Load balancing:** Several SRCs provide a full-flesh metadata management system. Queries from the SRCNet are redirected to the relevant servers taking into account the network connectivity (in general due to geographical proximity) or the load of the different servers. As in the previous case, every SRC could also have local high availability or load balancing.

Load balancing is difficult to implement for tables that are changing too fast as the synchronisation time could affect the database content. This is why a typical solution for load balancing is to create sticky sessions for the ones that are modifying database content, usually related to the creation of user tables in their private database schemas.

It is important to remark that the synchronisation of the servers could be done in different ways, either by direct database replication methods or through the implementation of ad-hoc database synchronisation services that could guarantee a more efficient way to consistent intermediate states. Also, server content should be properly backed up to allow disaster recovery.

The data model behind the metadata information must be shared between all the SRCs of the SRCNet and properly captured in the documentation. That implies that a common data model should be used. For scientific entities, there are several standards like CAOM (Dowler et al. 2008) or ObsCore (Louys et al.

2017). These are very well-defined data models that could be extended for particular observatories/missions.

Other data models needed to be standardised are the ones related to provenance and, in particular, the characterisation of scientific workflows or the ones used in product generation.

6.5. Authentication, Authorisation and Accounting

The SRCNet must implement a federated Authentication and Authorisation Infrastructure (AAI). This will integrate national federations through an international inter-federation service (e.g. eduGain) to enable the use of existing institutional accounts, and to allow the use of existing institutional credentials to authenticate with the SRCNet Infrastructure. The AAI will link these credentials to a centrally coordinated unique SRCNet Identity (SKA-ID?). A network of coordinated services will manage group membership and other relevant attributes, facilitating authorisation decisions for access to SRCNet data, computing, and other resources.

This solution should be implemented in a manner which may also make relevant decisions based on Accounting (usage and quota) information. The obvious requirement is to give users a view of their available quotas in advance, so they can decide whether to free up storage, ask for more computing, switch to a different group with more resources, or just submit their work knowing it should fit. Of course with shared resources, it is hard to guarantee that resources remain available during execution of a job without making the infrastructure more complicated.

Some software concerns for SRCNet AAI are:

- Users will be able to Authenticate to a single SRCNet identity, which may be linked to multiple different authentication identities. A typical example here may be a researcher's home institution, which is accessed via an international inter-federation service and the national identity federations. This means that user credentials, including passwords, do not need to be directly managed within SRCNet AAI and end services.
- Authorisation mechanisms will need to be implemented to control access to digital products and resources. This mechanism will need to consider either identity attributes or capabilities assigned on a job-by-job basis.
  Authorisation should be handled as transparently for the user as possible, using mechanisms such as credential delegation through tokens, so that different non-interactive modules can integrate seamlessly.
- As the SRCNet API should include AAI and information should be exchanged in an interoperable format (e.g. using AARC-G069 and similar) using standard schemata like inetOrgPerson, eduPerson, voPerson, etc.
- Supporting the science work, users should have persistent globally unique identifiers which are linked to - but different from - their home identity principal (so it can move with the users when they change home organisations, and/or be linked to multiple accounts).

Some software concerns for SRCNet Accounting are:

- The compilation of metrics like quotas or resources used should be compiled by the monitoring system (using a hierarchical approach) and stored in the Accounting system so, at least, a global resource allocation per user (and/or group) should be maintained in near-realtime. Local quotas or local resource allocation could be also set up, in particular for special resources only present in a limited number of SRCNet nodes.
- Users should be able to request modifications of their quotas (on users or groups) or resource allocations by using the helpdesk system. These requests should contain a description of the use case to be executed to justify the modification of the default values.
- The quota management system must not be confused with authorisation. The authorisation system is not clairvoyant: it does not know how many resources the user needs. Failure of authorisation should lead to an access denied error. Exceeding quota limits should lead to a failure of writing data or running jobs (or a warning, depending on whether the limits are hard or soft).

6.5.1 AA Interface possible approaches

The SRCNet AA interface could be implemented by using, at least, two approaches.

The first one, in the BPA sense[9], is to implement the global SRCNet AAI making use of a global proxy:

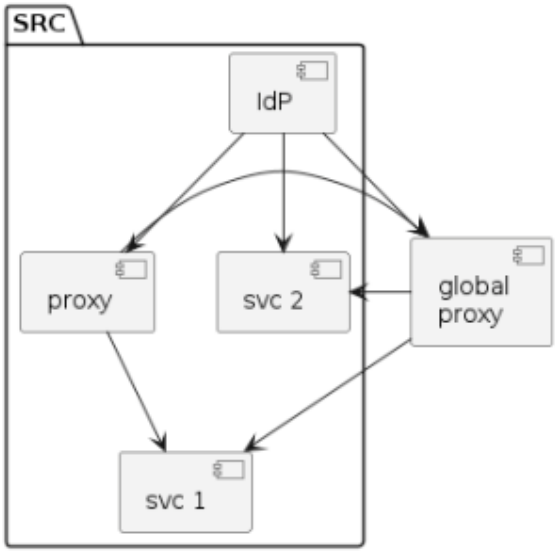

Figure 15. AA Interface using a global proxy.

The diagram shows an SRC in which IdPs can authenticate to SRC-based services directly or through an IdP/SP proxy (such as AUSSRC's Keycloak). Once the SRC is connected to SRCNet (the global proxy), users can authenticate to resources in (other) SRCs through the global proxy. Users can authenticate to the SRCNet proxy directly or through their SRC proxy; in turn, the identity they have through the SRCNet proxy gives them access to resources pledged by

[9] https://aarc-community.org/architecture/

the SRC to SRCNet (obviously for a user accessing resources within their own SRC, the global proxy should not be necessary).

The *principal* is the user's primary identifier, as seen through an authentication through a particular IdP (or proxy). Thus, in general, the user's principal is different depending on whether they authenticate directly with their IdP or they go through a proxy:

1. IdP→svc
2. IdP→SRC proxy→svc
3. IdP→global proxy→svc
4. IdP→SRC proxy→global proxy→svc

All of these authentication paths could potentially give the user different principals, though a persistent identifier like email, ORCID or an ePUID could be carried through. Having different principals is sometimes desirable, as users may change their home organisations but keep the proxied principals, or they would get different rights to access their own SRCs resources - accounting for resource usage would be different.

If the SRCNet implementation does not use global proxies, the SRCNet nodes would connect as follows:

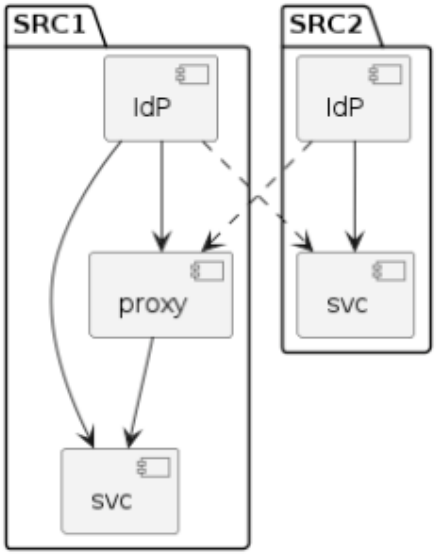

Figure 16. AA Interface not using a global proxy.

with SRC1 having its SRC proxy, and SRC2 not having one. It is possible to have SRC2's IdP cross authenticate to SRC1's proxy, as the proxy is built to handle multiple IdPs including the selection process. However, the trouble comes when we connect SRC1's IdP to SRC2's services: the service will now have multiple sources of identity publishing potentially different profiles for their users and potentially different protocols. This approach would vastly complicate running services. The implication is that every SRC will need to run its own proxy connected to the international inter-federation service.

Second approach implies a more independent SRCNet authentication system but technically more complex so the final implementation will result in the balance of political and technical aspects.

6.6. Collaborative environment

As the data produced by SKA will be massive for many scientific analyses, the users should be prepared to execute most of their science use cases in the SRCNet. As a requirement for the science produced, these analyses should be reproducible and, as a requirement of the scientific process, the users should be able to share and collaborate with their colleagues in the SRCNet.

Some functionalities should be implemented to create a collaborative environment:

- Users, teams and projects (based on groups) should have their distributed storage areas where intermediate and final digital objects could be stored close to the SRCNet data lake nodes and close to the SRCNet computing resources, preventing the need for upload/download of these products from the user's local environment during the analysis.
- Digital objects could be of different natures like data files, workflow descriptions, software containers, notebooks, tables, etc. All these objects should have a unique identifier in the SRCNet and a way to access them.
- Users' digital objects should be shareable by the users to other users or groups of users, allowing collaborative work.
- Users' digital objects could be flagged as public (shared with all).
- Different kinds of quotas should be associated with the different types of resources, to control the proper use of the SRCNet. These quotas could be modified at a user level by the operations team after a justified submitted request from users.
- Synchronisation of these digital objects in different locations (SRCNet nodes) would be needed in certain cases (e.g. software containers, tables, etc.) but this will follow a similar synchronisation procedure to the public general digital object repositories.

6.6.1. User storage areas

One of the required functionalities of the SRCNet is the possibility to execute workflows in the network itself. These workflows are planned to be executed as close to the data as possible to prevent latencies in accessing the data, either by a preliminary movement of data to prepare the environment or by using a streamed data approach.

However, although this has been described in other parts of this architecture, all these workflows would require in many cases input files from the users that are not located in the data lake and the executions will provide intermediate and final data products that need to be allocated at the server side. Also, these data files could be analysed as part of a collaboration so users should be able to share (with different access levels) with other users or groups within the network. Apart from the data files, other digital objects (e.g. software containers, workflows, notebooks, etc) would require similar levels of sharing options.

In this section, we will concentrate the analysis on the user storage areas, considering only the digital objects that can be serialised as objects/files into a storage repository.

The user storage areas will need to be integrated with analysis libraries to be used in science analysis threads. As many of the typical analysis libraries require to have the files used during the processing as a POSIX file system, the easier way to integrate storage elements ready to be used by these analysis libraries would be by using a software interface that could allow the mounting of these user spaces into the analysis environment although other more performant approaches will be studied. User storage areas can be abstracted with REST interfaces, as done by using the IVOA VOSpace protocol.

There are some aspects to be considered:

- User storage could/should be separated from the storage elements of the data lake to prevent security access problems
- User storage could be only present in one particular SRC node (the one associated with the user) and accessed from other SRC nodes computing resources in a stream-like way. This could be done if the data products inside the user storage areas are small enough to not be affected too much by the streaming latency
- User areas would provide an access to the data interface as closely as possible to the one used to access files in the data lake so single client access library can be used to access any kind of data from the SRCNet (user area and data lake products)
- Special methods to modify the visibility of data products at the level of read/write/execute for other users and groups

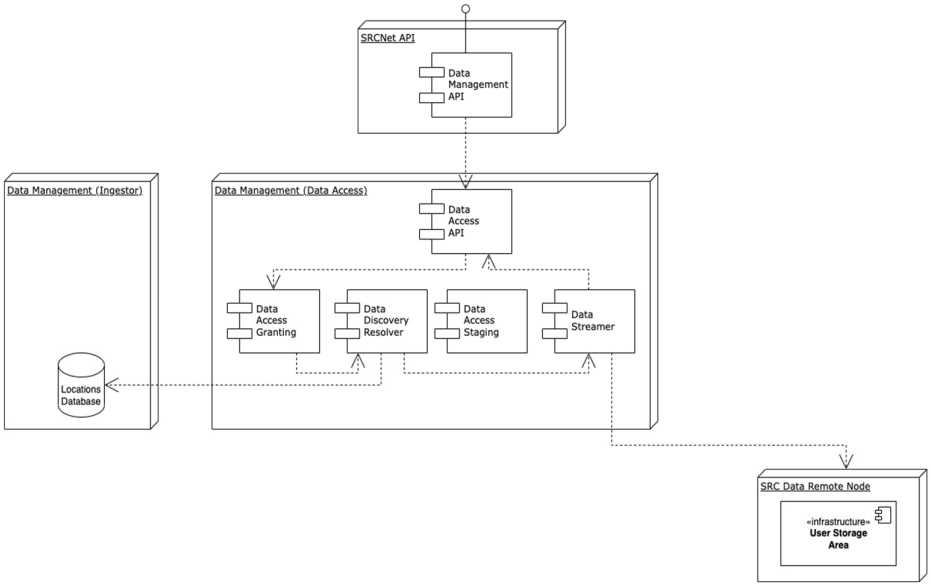

Figure 17. Diagram showing the access from data of one User Storage Area.

In this particular case, user storage data locations could be also present in the Locations Database so the data can be discovered. However, in most cases, the data will be accessed using a streamed protocol from a remote location. In case the data is too big to work efficiently in stream mode (which usually should not be the case for users' data, data replication to an area close to the user (to the invoking SRC node) could be also executed as done for general data lake products.

https://confluence.skatelescope.org/display/SRCSC/User+Storage+Areas

Users and groups are handled by the User Management System that should allow the discovery, modification of details and creation of new groups through a user interface and command line libraries.Users and groups are handled by the User Management System that should allow the discovery, modification of details and creation of new groups through a user interface and command line libraries.

Discovery from data from other astronomical missions/observatories could be done by using their Virtual Observatory protocols, allowing the saving of results from queries and retrievals into the User Storage Areas so science use cases that require the combination of data from different sources (e.g. multiwavelength astronomy, multi-messenger astronomy, etc.) could be implemented.

As data produced, intermediate products, or input files could be quite big and taking into account the expected high number of scientists making use of the SRCNet, the need for quotas on the User Storage Areas, at the user (individual users and projects/groups) level, would be a must. In case the quota is exceeded during one workflow execution, a quota exceeded should be sent as a result and the process, if needed, should be aborted.

If for a particular science use case or program, a temporary increment of the quota is needed, a helpdesk request should be raised, analysed and assessed by the SRCNet operations team and, if accepted, implemented at the user level (including the possible restoration to the original value after the specified period).

For some high-intensive science use cases, data from the data lake could be required to be moved to a buffer area close to the computing resources (in particular when a particular remote operation on the data could not be done or is not supported from the remote operations interface). This buffer area could not be, in principle, the same storage area as the one used by the User Storage Areas (probably it is more convenient to have different storage elements for this quite volatile location). It is to be decided how these buffered data lake product replicas compute on the user's quota (either by counting in their quota, counting in a separate quota or not counting at all in the user's quota). This would be decided in coordination with the SRCNet node members.

6.6.2 Persistent Table Upload

There is a set of science use cases that require access to a combination of tables accessible by the user (public and science project tables that require special privileges) and tables uploaded by the users. This is a known problem in astronomy that, for example, users have to address operations like astronomical sources cross matches from tables produced by the project a users' source catalogues.

To solve this problem in a clean way, some solutions were developed like traditional TAP (Dowler et al.) where a REST interface is defined that allows volatile uploads of tables before a certain query execution, allowing the use of this volatile table as part of the query.

However, this kind of approach is not efficient for certain use cases as it requires the streaming of the table from the user area before every query. Also, it does not allow the sharing of users' tables with other users and groups of users to allow collaborative work.

In order to solve this problem, some solutions were defined, e.g. the ESA TAP+ (Salgado et al. 2017) or CADC (Dowler and Major 2018).

To allow this, the following elements need to be implemented:

- There must be defined, at least, one credentials-handling mechanism declared for the service so the client could use it after discovering. This is covered by the new IVOA TAP 1.1 standard (Dowler et al. 2019)

- A discovery service needs to be implemented to show the tables available to the user. For non-authenticated users, this will only show public tables. For authenticated users, the output of this discovery service will also show extra tables accessible to the user.
- The query parsing module should check that the tables present in the query allow accessible tables and return an error when a table present in the query is not accessible for the user, in a similar way that an error should be raised for tables not present in the system. To guarantee privacy on the users' schema, the error could be the same.
- A REST or equivalent service should be implemented to upload tables to the users' database schema. This service could also allow the creation of indexes on the different columns or geometrical indexes on geometrical columns for future good performance.
- The total size of the tables present in the users' schema should be controlled by quotas. Uploading exceeding the quota would be denied.
- There should be a way to request and modify users' schema quotas for users by the operations team.

6.7. Computational resources allocation and federated execution

The SRCNet should allow the federated exec of processes as close as possible to the data, using the best possible load-balancing of the resources available, including real-time information on the status of the resource, and deciding the execution of workflows on the most convenient computational resources (e.g. HPC resources).

This is why the need for a federated execution requires global/common services that make use of the description of the SRCNet topology (resources available per node, the status of the network, load and status of the resources, location of the input data, authorisation rules, evaluation of best possible execution plan, etc).

Metadata describing the SRCNet topology should be provided by the relevant SRCNet nodes by providing events to global services (e.g. a federated event bus) using a hierarchical approach so the information can be compiled and used by a global Execution Scheduler. As said, the information obtained hierarchically from all the nodes will be an aggregated topology (probably inside a database in which content changes in real-time). As ensuring synchronised content for all the instances could require a heavy synchronisation process, having only a limited set of full implementations would be recommended. This is why this service is considered global. As a global service, this service could be deployed only in a certain number of SRCNet nodes (SRCNet node to decide) so other SRCNet nodes could make use of it for their operations (e.g. by the shared execution planner).

This submodule, part of the Workflow Management module, would be used to decide on a good execution strategy on the SRCNet to prevent latency and good performance metrics.

Once the best possible strategy for execution is identified, the global/common workflow management system should request the execution of the process (either by a direct API invocation (active) or by using a federated event bus request to the required SRC node(s)).

In both cases, information on the federated execution will be provided by the execution SRC node(s) to the monitoring management system to update statistics, update user quotas and manifest status among others.

Shared software should be in a portable format so local software repositories should be synchronised/federated. In this way, execution processes and libraries are usually present close to the execution framework preventing latency. The SRCNet will provide templates of software portable formats and access libraries to optimise their execution into the SRCNet architecture.

Locally, orchestrators should allow the spawning of the required execution processes to obtain local software resilience. Global/common workflow management system and the monitoring status ensure global software resilience.

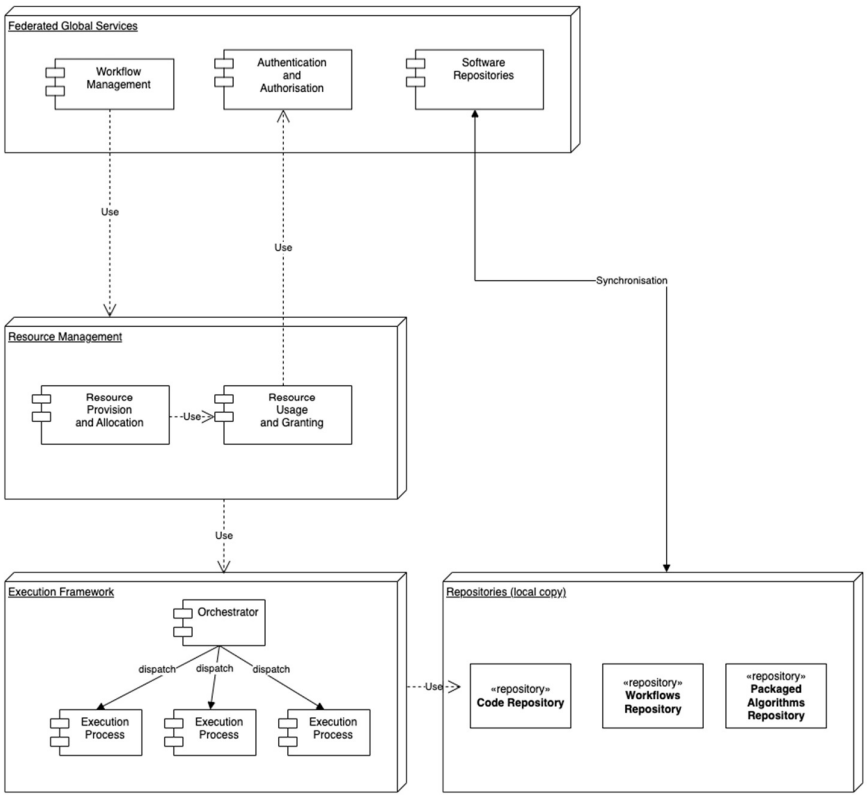

Figure 18. Diagram showing the federated execution of one process.

A global workflow management system, using a global execution scheduler that makes use of one workflow analysis system and real-time information, decides an executor SRCNet node(s). Locally, the execution framework contains an orchestrator that uses a local repository synchronised copy of the software for execution.

https://confluence.skatelescope.org/display/SRCSC/Federated+Execution

For simple cases of accessing data through remote operations, the data mesh approach is described in 6.2.2. Data mesh approach could be used. In this case, there would not be implemented a full-fleshed federated execution but the instantiation of parsing libraries services close to the data and the gathering of the result of the operations in remote SRCNet nodes into one processing SRCNet node where the user

is declaring and executing the workflows. This approach is lightweight and easy to implement in many cases with the following exceptions:

- Data Mesh is mainly applicable to direct operations on the data. Complex operations or workflows that require data aggregations are not easily mapped with this approach.
- Only operations that have been declared, implemented in the parsing libraries and distributed are available. New operations needed for a particular science use case require implementation and distribution. This second exception could be mitigated if the operation is declared by the user, converted to a portable software (e.g. by creating a container), distributed to the remote SRCNet node and executed by a remote orchestrator but the performance of the operation and the correct execution is not guaranteed as the default official operations declared in the official libraries. Also, users portable code could be prone to security analysis by the SRCNet Operations team and software to detect anomalous behavior and breaches, although the approach to be generally used is to provide portable software templates to be used by users with some security rules already in place.

6.7.1. Shared Execution planner

Integrating heterogeneous hardware resources for execution on the SRCNet would require, as commented, the implementation of a shared execution planner. This shared execution planner should have metadata available to decide on the best possible location(s) where a particular workflow, e.g. a science analysis thread, should be executed.

To obtain this information, several aspects are required, e.g., the metadata gathering from the resources, the location of the input data to be used, and real-time information on the status of the network.

In the case of resource characterisation, a unique metadata format would be required to compile the available hardware resources (sometimes heterogeneous) with comparable metrics. Some metadata definitions have been done for this particular problem, mostly focused on the cloud hardware metadata characterisation.[10] [11]

The SRCNet topological model needed should define different aspects to be captured in an abstract but homogeneous way resource aspects like Scalability, Autonomy, Availability, QoS, Performance, Consistency, Security and Reliability.

On computing non-functional requirements, metadata can be divided into metrics like (non-exhaustive list):

- Computing engines:

[10] mOSAIC ontology
https://www.researchgate.net/publication/220726510_An_Analysis_of_mOSAIC_ontology_for_Cloud_Resources_annotation

[11] TOSCA http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html

    - CPU: CPUSpeedProperty, CPUNumberOfCores, CPUArchitecture, CPUTypeProperty and CPUFlopsProperty, etc
    - GPU: GPUSpeedProperty, GPUNumberOfCores, GPUArchitecture, GPUTypeProperty and GPUFlopsProperty, etc
- Memory:
    - MemoryAllocationProperty (to capture Policies in the memory allocation), MemorySize (total size), etc
- Network:
    - NetworkLatencyProperty, NetworkDelayProperty, NetworkBandwidthProperty, etc
- Data Lake:
    - ResourceElementSpaceProperty, ResourceElementOccupancyProperty, ResourceElementTransferRatePropety, etc
- General node/cluster metadata:
    - OccupancyProperty, AvailabilityFlag, etc
    - NetworkLatencyA, NetworkLatencyB, etc

At the same time, the characterisation of the software elements of the workflow to be executed is required in order to define the dependencies of a particular software instance that are needed for the execution:

- Software versioning, availability and dependencies
- Data Dependencies:
    - PID (persistent identifier) of input data to resolve location, size and type
    - Software Architect dependencies:
    - SupportForCPU, SupportForGPU, SupportForFPGA, SupportForARM, etc
- Computing engines dependencies:
    - NumberOfCPUs, NumberOfGPUs, CPUFlops, GPUFlops, etc
- Memory Requirements:
    - MemoryRequired
- Priority Requirements:
    - InteractiveFlag, BatchFlag, PriorityFlag, etc

All these metadata should be gathered by the monitoring system by using different possible approaches:

- Centralised approach: All metadata is gathered in a central repository
- Decentralised approach: Metadata is gathered into the different SRCs and it is made available for the different SRCs
- Hierarchical approach: Similar to the decentralised approach but allowing different internal levels at every SRCs

A similar approach could be used to include pure services into the SRCNet providing a software description of the functionality and performance indicators.

Resources, software and services characterisation metadata are available for discovery and detailed queries through the Information System. The shared Execution Planner submodule inside the Resources Allocation system would use all these metadata to identify a good candidate for the execution of a certain workflow within the SRCNet.

A future upgrade of the shared execution planner could imply the use of heuristic and machine learning algorithms to optimise the result using the real metadata on the use of the system, although these algorithms should take into account the unavailability of resources through real-time monitoring status.

6.8. Distributed software management

The SRCNet will need to deploy, federate, and synchronise distributed software across the SRCNet nodes, making sure that all dependencies are satisfied for each version deployed. The main categories of software are:

- Common API interfaces: Common modules defining API signatures will be defined and propagated to the different SRCNet nodes to ensure harmonised behaviour. The update and deployment of these modules, which could have different local implementations, must be coordinated and planned so local implementations could be also updated. The same coordination process will be used for other central elements of the science application or libraries.
- Services software: Core modules will be defined and will be quite standard for the SRCNet to ensure that some of the basic functionalities have the same behaviour in different SRC nodes. For example, the client and server libraries of some of the management system modules (metadata and data access, federated computing execution, authentication/authorisation rules checks,...) should be common to guarantee security
- Infrastructure software: Infrastructure provisioning will look different at the SRCNet nodes owing to their heterogeneity, however by splitting areas of concern vertically in the stack and establishing APIs at the different layers, sites will be able to share and collaborate on building infrastructure in a consistent manner whenever possible. This may include deploying an orchestrator management solution to consistently deploy services software. Defining the required infrastructure as code will also help new, less-experienced sites get started in deploying the SRCNet services (software category above) in minimal time. Infrastructure as Code (IaC) is an extension of the DevOps methodology and is a means by which infrastructure is provisioned and managed using code instead of manual processes. This allows for better resiliency, consistency, collaboration and increased speed to deployments of services. IaC will serve as a starting point to build a community for SRCNet Operations.
- Science software: Science workflows will make use of libraries that do not only require to be accessible in all the SRC nodes but, also, it would be required to have specific versions and other relevant software metadata (e.g. target architecture, properties, etc) to ensure reproducibility. Also, the science workflows and elements created by the users could need to be available for execution in all the SRCNet nodes.

There are different technical solutions to enable this functionality in the SRCNet but two aspects should be considered in the design:

- Commonality across the software categories would be in the form of a shared software repository created to have a common and consistent repository of science analysis libraries, SRCNet science platform modules and, in the collaboration side, science workflows created by the user. This shared software repository could be federated and have different instances synchronised at different time zones to provide high availability and, if required, load balancing.
- Software lifecycles, associated policies and metadata characteristics will look different for the different types of software and thus these types of software will be treated differently during the building, storing and distribution phases. Building portable software will be essential for all software categories but especially for services and infrastructure software. Artefact registries can be used for storing and distributing services and infrastructure software with the corresponding metadata required for versioning, however, metadata needs may entail a separate entity to

manage artefact metadata in an efficient manner. Older software versions could need to be available, in particular the ones referenced in workflows, in order to facilitate reproducibility.
- API documentation should consider migration, replication and upgrade processes, particularly with respect to infrastructure software.

The shared software repository should allow flagging the versions to be instantiated by default of the different modules, to guarantee that the same version of the SRCNet core modules is running in all the SRCNet nodes and to update the global network when a new version of one particular module is set to the default version. Also, that could require that the orchestrators in charge of the execution of the SRCNet science platforms nodes are refreshing and react to these flags. This synchronisation of modules across the SRCNet could be obtained using a common CI/CD tool and setting the modules to the *default version*, including other dependent modules and external dependencies that should be also updated to prevent incompatibilities.

Software is only valuable as long as it is maintained. It will also be important to consider the maintainers of (all) the types of software and establish communities for short-term or long -term maintenance as applicable. This may be by identifying maintainers within the SRCNet community, and/or considering applications of our software to other communities and sourcing person-power from the open-source community.

Appendix 1: Requirements Supporting Architecture Principles

In the following subsections, we present SRC Level 0 requirements supporting the different Architectural principles defined in the previous section. These requirements were compiled and curated by the different working groups of the SRCNet.

A1.1 The main objective of the SRCNet is to maximise the science produced by the community using SKA data

Key	Summary
SRC-20	SRCNet Ingest of SKAO Data Products
SRC-27	Multi-wavelength support
SRC-32	Facilitation of collaboration for data post-processing and analysis
SRC-77	Mapping Science team needs to SRC resources

SRC-96	Baseline Implementation Deadline
SRC-102	Science analysis platform
SRC-103	User Proficiency in Radio Astronomy
SRC-112	GUI and CLI interfaces
SRC-117	DOIs of publications related to archive products
SRC-134	User Support and Help Desk
SRC-155	Authorised Programmatic Access to the Science Archive Data Products
SRC-189	User-configured pipelines
SRC-190	Include custom software in workflows
SRC-199	Document publicly-accessible capabilities
SRC-202	Save User Queries for the Science Archive
SRC-209	SRCNet GUI Accessibility

A1.2 SRCNet development is a global effort done by all the SRCs

Key	Summary

SRC-50	Coordination of resources and services
SRC-71	Monitoring SRCs against MoUs

A1.3 Architecture should be scalable

Key	Summary
SRC-14	Overall SRC storage capacity must be scalable
SRC-19	Global Data Management System
SRC-29	Software Interoperability
SRC-71	Monitoring SRCs against MoUs
SRC-78	Interoperability
SRC-142	Portable Computer System Provisioner Service
SRC-144	Software-Defined Infrastructure
SRC-207	SRCNet Scalability

A1.4 Architecture should be extensible

Key	Summary

SRC-33	Common API and CLI
SRC-42	User applications in SRCNet Toolbox

A1.5 Data and Computing Resilience

Key	Summary
SRC-17	Data Availability in pledged storage
SRC-19	Global Data Management System
SRC-21	Data Integrity is assured during replication
SRC-46	Realtime status
SRC-48	Monitoring Suite
SRC-50	Coordination of resources and services
SRC-61	Science Data Products Publishing
SRC-71	Monitoring SRCs against MoUs
SRC-72	Graceful exit from the SRC Network

SRC-109	Availability of Science Data Product Index system
SRC-135	Centralised monitoring of services
SRC-153	High Availability of Data Management Service
SRC-197	Quality Assurance for User Software
SRC-206	SRC Net Security
SRC-208	SRCNet Reliability
SRC-232	Global Network Health View

A1.6 FAIR principles

Key	Summary
SRC-23	Data provenance repository
SRC-38	Data stewardship in software and frameworks
SRC-40	Interoperability of Data Formats
SRC-41	SRCNet Abstraction and reusability

SRC-53	IVOA Access Services
SRC-54	Virtual Observatory Metadata Models
SRC-55	Public API Protocol
SRC-56	Public API Standards
SRC-61	Science Data Products Publishing
SRC-63	Browsable Science Data Products Catalogue
SRC-65	Observatory Data Products Findability Metadata
SRC-66	New Products Registration
SRC-80	Data Products Findability
SRC-81	Advanced Data Products Metadata Association
SRC-84	Archived Data Products Accessibility
SRC-88	Advanced Data Product Documentation
SRC-90	Data Access Policy

SRC-115	Public data findable through the search service
SRC-118	Uniform vocabulary in the archive
SRC-167	Use of IVOA recommendations for ADP (meta)data
SRC-168	FAIR compliant vocabularies
SRC-169	Qualified references to other (meta)data within ADPs
SRC-171	Access to the ODP metadata when ODP no longer available or not authorised access
SRC-172	ODPs with accurate and relevant attributes following IVOA standards
SRC-173	ODPs with a clear and accessible data usage license
SRC-193	Associate DOI to output data
SRC-198	Exchange Archive Products with External Institutions
SRC-211	Software and container repositories interface
SRC-216	Persistent links to data products.

SRC-271	FAIR compliance and proper credit

A1.7 The SRCNet Architecture should optimise data logistics, preventing unnecessary moving of data

Key	Summary
SRC-15	SRC Data Index
SRC-19	Global Data Management System
SRC-26	Policy driven data distribution and processing
SRC-74	SRC - SRC interfaces
SRC-136	Data visualisation service
SRC-139	SRCNet shall have federated data management/location service
SRC-158	SRC Data Management Data shall allow replica requests
SRC-185	Access multiple distinct inputs from within a workflow
SRC-194	Provide “scratch space” for use during workflows
SRC-226	Managing data transfers

A1.8 SRCNet should allow federated execution

Key	Summary
SRC-29	Software Interoperability
SRC-32	Facilitation of collaboration for data post-processing and analysis
SRC-114	Unified access and query interfaces for the Science Archive
SRC-143	Federated Services for Compute/Workflow Management

A1.9 Execution of analysis workflows should be reproducible

Key	**Summary**
SRC-22	Reproducibility of results
SRC-23	Data provenance repository
SRC-24	Software versioning
SRC-30	Software lifecycle management
SRC-85	Provenance Repository Management
SRC-86	Provenance Repository Content
SRC-87	Provenance Capture

SRC-88	Advanced Data Product Documentation
SRC-94	Metadata management for acceptable Science Archive products
SRC-105	Consistent Release Status
SRC-138	Provenance Visualization
SRC-141	Distributed Software/Workflow Repository
SRC-174	Associate the ODPs with detailed provenance
SRC-181	Create custom pipelines and configurations

A1.10 Curation and preservation

Key	Summary
SRC-24	Software versioning
SRC-30	Software lifecycle management
SRC-31	Software and container repositories
SRC-106	Long-term preservation of Observatory Data Products
SRC-107	Long-term preservation of Advanced Data Products

SRC-110	Search web service to query archive products
SRC-140	IVOA-Compliant Data Archiving
SRC-204	Data Releases
SRC-212	Software and container repositories policies

Appendix 2: Actors fine-grained classification

A2.1 Developer role fine-grain classification

Fine-grained classification of the Developer role extracted from specific use cases.

Role	Description	Basic Permissions
Application Developer	Someone who is developing user front-end functionality of SRCNet tools.	Advanced Execution Upgraded Quotas Software Publishing Deployment
Component Developer	Someone who is developing components of software to be used in workflows that are to be executed in SRCNet.	Advanced Execution Upgraded Quotas Software Publishing Deployment
Pipeline Developer	Someone who is developing data processing pipelines and scientific workflows that are to be executed in SRCNet.	Advanced Execution Upgraded Quotas Software Publishing
SRC Developer	Someone who is affiliated with one of the SRCs or SKAO and developing software for the core functionality of SRCNet.	Advanced Execution Upgraded Quotas Software Publishing

A2.2 Scientist role fine-grain classification

Fine-grained classification of the Scientist role that could be used in specific use cases.

Role	Description	Basic Permissions
Researcher	Someone who is using SRCNet to work with the SKAO data, and is affiliated with a research institution.	Advanced Execution Extended Quotas
Citizen Scientist	Someone who is not affiliated with a research institution but wants access to SKAO data.	Restricted Execution Restricted Quotas
Commercial user	(Someone from...) A company that would like to work with SK AO data.	Advanced Execution Restricted Quotas
Principal Investigator	Someone who leads a science project/proposal.	Advanced Execution Upgraded Quotas
Project Manager	Someone who manages a team of scientists working with the SKAO data using SRCNet.	Advanced Execution Upgraded Quotas
Public Data Researcher	A researcher with generic access to low-cost public services, e.g., query public metadata, visualize public data, request cutouts of public data, and download a reasonable amount of public data.	Restricted Execution Restricted Quotas
Public Data Researcher with resource allocation	A Public Data researcher with authenticated access to processing and storage resource allocation.	Restricted Execution Extended Quotas
Science Project Member	Someone who is using SRCNet resources allocated to a science project.	Advanced Execution Upgraded Quotas
SKAO Staff Researcher	An SKAO researcher with authorized access to all or parts of the data collection and processing and storage allocation in their role as support astronomer.	Advanced Execution Upgraded Quotas Superuser access

SRC Coordination Committee (SCC) Member	A member of the SRC Coordination Committee (high-level SRC project oversight committee).	Advanced Execution Upgraded Quotas Superuser access
Support Scientist	Someone who is affiliated with one of the SRCs or SKAO and assists the SRCNet users with the science tasks through the helpdesk.	Advanced Execution Upgraded Quotas
Community engagement/ Professional Research Investment Manager (PRISM)	Someone who supports the growth of users and communities on the facility. They contribute to maximising the science and value of research investments.	Advanced Execution Upgraded Quotas

A2.3 Operator role fine-grain classification

Fine-grained classification of the Scientist role that could be used in specific use cases.

Role	Description	Basic Permissions
SRC Operations Group Member	A member of the SRC Operations Group, which will track day-to-day global SRC system health, and follow up on reported problems to ensure that issues are resolved. The members of this team should be able to perform corrective procedures into the system and provide stats, metrics and produce dashboards on the SRCNet usage. The SRC Operations Group will be composed of members from different SRCs and SKAO so that both remote and local operations could be executed. Also, this multi-SRC composition will help to have better maintenance support at different hours.	Advanced Execution Upgraded Quotas Superuser access

SRC Site Operative	An operative working at an individual SRC site (operating and maintaining HW and per-SRC services).	Advanced Execution Upgraded Quotas Local admin priviledges
Technical support operator	Someone who is affiliated with one of the SRCs and assists the SRCNet users with technical tasks through the helpdesk.	Advanced Execution Upgraded Quotas

References

The following documents are referenced in this document. In the event of a conflict between the contents of the referenced documents and this document, **this document** shall take precedence.

Breen, S., Bolton, R., & Chrysostomou, A. (2021, May 18). *SKAO SCIENCE DATA PRODUCTS: A SUMMARY*. Retrieved May 3, 2023, from https://aussrc.org/wp-content/uploads/2021/06/SKA-TEL-SKO-0001818-01_DataProdSummary-signed50.pdf

Clarke, A., Franzen, T., Breen, S., & Bolton, R. (n.d.). *SRCNet Use Cases*. SRCNet Use Cases. https://docs.google.com/document/d/12Tlg438xfZahNusCfAdqzm6u6XqUZQ8HdhtgI5zr5rQ/

Dehghani, Z. (2022). *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media, Incorporated.

Dowler, P., Gaudet, S., Duran, D., Redman, R., Hill, N., & Goliath, S. (2008). Common Archive Observation Model. In R. W. Argyle, J. R. Lewis, & P. S. Bunclark (Eds.), *Astronomical Data Analysis Software and Systems XVII: Proceedings of a Conference Held in Kensington Town Hall, London, United Kingdom, 23-26 September 2007*. Astronomical Society of the Pacific.

Dowler, P., & Major, B. (2018). *YouCat: User Catalogue Service*. IVOA Nov 2018 DAL session. Retrieved 03 14, 2023, from https://wiki.ivoa.net/internal/IVOA/InterOpNov2018DAL/tap-youcat.pdf

Dowler, P., Rixon, G., Tody, D., & Demleitner, M. (2019, September). *Table Access Protocol Version 1.1* (1.1). Table Access Protocol Version 1.1. https://www.ivoa.net/documents/TAP/20190927/

Franzen, T., Clarke, A., Breen, S., & Bolton, R. (n.d.). *SRCNet Visualisation Use Cases*. SRCNet Visualisation Use Cases. https://docs.google.com/document/d/1gnPBvzAi9hOkS5zCeUiGkdhwQGZepZbe/

Gorelik, A. (2019). *The Enterprise Big Data Lake: Delivering the Promise of Big Data and Data Science*. O'Reilly Media.

Louys, M., Tody, D., Dowler, P., Durand, D., Michel, L., Bonnarel, F., & Micol, A. (2017, May). Observation Data Model Core Components, its Implementation in the Table Access Protocol Version 1.1. *IVOA specifications*. 10.5479/ADS/bib/2017ivoa.spec.0509L

Quinn, P. (n.d.). *SRC White Paper v1.0 Final+*. Retrieved May 3, 2023, from https://aussrc.org/wp-content/uploads/2021/05/SRC-White-Paper-v1.0-Final.pdf

Quinn, P., Axelrod, T., Bird, I., Dodson, P., Szalay, A., & Wicenec, A. (2015). *Delivering SKA Science in Proceedings of Advancing Astrophysics with the Square Kilometre Array*. SKAO. 10.22323/1.215.0174

Salgado, J., Bolton, R., Swinbank, J., Joshi, R., Sánchez, S., Villote, J., Gaudet, S., Yates, J., Barbosa, D., Taffoni, G., Bradley, F., & van Haarlem, M. (2023). *SRC Net Top-Level Roadmap*.

Salgado, J., Gonzalez-Nuñrez, J., Gutierrez-Sanchez, R., Segovia, J. C., Durán, J., Hernández, J. L., & Arviset, C. (2017). The ESA Gaia Archive: Data Release 1. *Astronomy and Computing*, *21*(1), 22-26. https://www.sciencedirect.com/science/article/abs/pii/S2213133717300355

Simmonds, R. (n.d.). SKA-SDP Memo SKA-TEL-SDP-0000060 v01C. https://ska-sdp.org/sites/default/files/attachments/ska-tel-sdp-0000060_01c_rep_sdpmemoregionalcentres_-_signed.pdf

SKAO. (2020). *SKA Observatory Establishment and Delivery Plan*. SKAO.

Wu, C., Wicenec, A., & Checcucci, A. (n.d.). Optimizing NGAS for the MWA Archive. *Experimental Astronomy*, *36*, 679–694. https://doi.org/10.1007/s10686-013-9354-1