UKSA CDSSG Task 2: System design for routine climate data processing at Jasmin-CEMS

Jane Lewis and Debbie Clifford
Document Control

Contributors

<table>
<thead>
<tr>
<th>Person</th>
<th>Role</th>
<th>Company</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>J. Lewis</td>
<td>System Engineer</td>
<td>UoR</td>
<td>Author</td>
</tr>
<tr>
<td>D. Clifford</td>
<td>Project manager</td>
<td>IEA</td>
<td>Review</td>
</tr>
</tbody>
</table>

Distribution

<table>
<thead>
<tr>
<th>Person</th>
<th>Role</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Clifford</td>
<td>Project manager</td>
<td>IEA</td>
</tr>
<tr>
<td>V Bennett</td>
<td>consultant</td>
<td>STFC</td>
</tr>
<tr>
<td>K Halsall</td>
<td>consultant</td>
<td>Telespazio</td>
</tr>
<tr>
<td>A Hayward</td>
<td>consultant</td>
<td>Telespazio</td>
</tr>
</tbody>
</table>

References

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Document</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>resc: 09/05/05/01_UKSA CDSSG_requirements_1.0</td>
<td>26-01-16</td>
<td>Requirements specification (issued)</td>
</tr>
</tbody>
</table>

Revision History

<table>
<thead>
<tr>
<th>Issue</th>
<th>Author</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0A</td>
<td>J Lewis</td>
<td>23-02-16</td>
<td>Initial draft from EA model baseline 03</td>
</tr>
<tr>
<td>1.0</td>
<td>D Clifford</td>
<td>10-03-16</td>
<td>Final version following internal review</td>
</tr>
</tbody>
</table>

Definition of Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>data retrieval</td>
<td>obtaining required dataset(s) from live or archive storage. In this engineering document, retrieval does not refer to the process of deriving geophysical information from a data stream.</td>
</tr>
<tr>
<td>&quot;Must&quot;, &quot;Should&quot;</td>
<td>terms used in the MoSCoW method of analysing requirements. The</td>
</tr>
</tbody>
</table>
MoSCoW method is a prioritization technique used to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement. Requirements labelled as MUST are critical to the delivery. Requirements labelled as SHOULD are important but not necessary for delivery. Requirements labelled as COULD are desirable but not necessary. Requirements labelled as WON'T (or "not needed") have been agreed by stakeholders as the least-critical, lowest-payback items, or not appropriate at that time.

**public domain**
the state of belonging or being available to the public as a whole, and therefore not subject to copyright. For software in particular there is no ownership such as copyright, trademark, or patent. Unlike other classes of licenses, there are no restrictions as to what can be done with the software.

**source control**
a component of software configuration management, source control is the management of changes to documents, computer programs, large web sites, and other collections of information.

**validation / verification**
software verification is ensuring that the product has been built according to the requirements and design specifications, while software validation ensures that the product actually meets the user's needs, and that the specifications were correct in the first place.

### Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JASMIN</td>
<td>[UK &quot;super-data-cluster&quot; which delivers infrastructure for data analysis]</td>
</tr>
<tr>
<td>CEMS</td>
<td>Climate and Environmental Monitoring from Space</td>
</tr>
<tr>
<td>IEA</td>
<td>Institute of Environmental Analytics</td>
</tr>
<tr>
<td>STFC</td>
<td>Science and Technology Facilities Council</td>
</tr>
<tr>
<td>UoR</td>
<td>University of Reading</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
</tbody>
</table>
# Table of Contents

Document Control .................................................................................................................. 2  
Contributors .......................................................................................................................... 2  
Distribution ........................................................................................................................... 2  
References ............................................................................................................................... 2  
Revision History ..................................................................................................................... 2  
Definition of Terms ................................................................................................................ 2  
Acronyms and Abbreviations ................................................................................................. 3  
Table of Contents ................................................................................................................... 4  
Introduction ............................................................................................................................ 6  
Scope ..................................................................................................................................... 6  
Overview ................................................................................................................................. 6  
Components ............................................................................................................................ 8  
  Component : Archive Creation & Maintenance .................................................................... 8  
  Methods ................................................................................................................................. 8  
  Contains class: ArchiveInputManager ................................................................................. 9  
Component : Archive Search & Extraction ......................................................................... 9  
  Methods ................................................................................................................................. 9  
  Contains class: ArchiveRequestController ...................................................................... 9  
Component : Data Ingest & Monitoring .............................................................................. 9  
  Methods ................................................................................................................................. 9  
  Contains class: DataIngestManager .............................................................................. 10  
Component : Environment .................................................................................................. 10  
  Methods ................................................................................................................................. 10  
  Contains: UserSupport ...................................................................................................... 11  
  Contains: Validation .......................................................................................................... 11  
  Contains class: AccreditationManager ........................................................................... 11  
  Contains class: VM_Management ...................................................................................... 11  
Component : Process Monitoring ....................................................................................... 11  
  Methods ................................................................................................................................. 11  
  Contains class: CoreMonitor ......................................................................................... 12  
  Contains class: MemoryUseMonitor ................................................................................ 12  
  class: DiskUseMonitor ..................................................................................................... 12  
  class: RAMUseMonitor ..................................................................................................... 12
Introduction
The Climate Data from Space Stakeholder Group (CDSSG) has a long term goal to develop and demonstrate a "seamless supply chain for climate data from space", exploiting existing infrastructure to develop better multi-layer, multi-data services.

The initial phase of work was to develop a series of case study reports on the use of climate data from space by the UK public and private sectors, which documented the limitations of the current supply chain in supporting those users. This second phase of work progresses the seamless supply chain by developing a professional system design. This uses input from the CDSSG commissioned implementation plans produced by EO researchers to show how their mature "essential climate variable" products could be implemented in an operational framework.

This document is for partial completion of the deliverable on Task 2: “system design for operational short-delay climate data record production at JASMIN-CEMS from Sentinel 3, as part of the "seamless supply chain" for climate data from space: phase 2 of 2015-16 work plan”. It follows on from the previous deliverable for this task, the Requirements Specification (reference [1]).

The purpose of this document is to describe a system which supports common tasks across the different essential climate variable products, promoting consistency and maintainability (as outlined in the requirements document). The document contains descriptions of the components, actors and use cases for the system. Annex A contains the traceability matrix between requirements and the components and use cases. Annex B contains use case sequence diagrams. Further design detail is contained in the UML model which is a Sparx Enterprise Architect .eap file, available on request.

Scope
There is no plan to substantially change the existing infrastructure or system architecture; there should only be additional components or services required. A major issue will be storage capacity for the data, with a need for a short time-delay archive retrieval system supported by an improved catalogue and search system. There may also be a need for more processing power within the cluster and measures to address any VM hosting capability limitations. Other constraints are likely to be support for data acquisition in terms of manpower to monitor and fix, and support for users to operate the system efficiently.

Overview
This section describes the components thought to be needed to implement the requirements. Note that the existing system may well contain components which are similar and/or have overlapping functionality. It is not intended that the new components replace the existing capabilities, but rather that they augment and integrate with the current system.

Operations on the components are derived from the use cases but not all are required to be implemented in code: many are process and/or user actions. Figure 1 illustrates the mapping between components and requirements: note that not all requirements are realised by system functionality.
1.1.1 the system must provide a mesh
(from 1.1 Feedback for issues/quality)

1.1.2 input data must meet quality
(from 1.1 Feedback for issues/quality)

1.2.1 the system must alert on input
(from 1.2 Monitoring input)

1.2.2 the system must alert on input
(from 1.2 Monitoring input)

1.2.3 the system should prompt process
(from 1.2 Monitoring input)

1.2.4 the system could provide an input
(from 1.2 Monitoring input)

2.1.1 the system must provide a mesh
(from 2.1 Archive retrieval)

2.1.2 the system must provide a file
(from 2.1 Archive retrieval)

2.1.3 the system should offer alternative
(from 2.1 Archive retrieval)

2.1.4 the system must make the job run
(from 2.1 Archive retrieval)

2.1.5 the system shall accept processing
(from 2.1 Archive retrieval)

2.1.6 the system shall accept processing
(from 2.1 Archive retrieval)

2.1.7 the system shall make the job run
(from 2.1 Archive retrieval)

2.2.1 output datasets must be a mesh
(from 2.2 Quality assurance of output)

2.2.2 the system shall provide a file
(from 2.2 Quality assurance of output)

2.2.3 the system must accommodate disk space
(from 2.2 Quality assurance of output)

2.3.1 the system must provide sufficient storage
(from 2.3 Storage needs)

2.3.2 the system must provide disk space
(from 2.3 Storage needs)

2.3.3 the system must accommodate data
(from 2.3 Storage needs)

2.4.1 the system must provide sufficient processing
(from 2.4 Processing needs)

2.4.2 the system must provide processing
(from 2.4 Processing needs)

2.4.3 depending on loading figures
(from 2.4 Processing needs)

3.1.1.a simplification of job submission
(from 3.1 User support for optimised processing)

3.1.1.b provision of a simple submission
(from 3.1 User support for optimised processing)

3.1.1.c provision of technical support
(from 3.1 User support for optimised processing)

3.1.2 the system shall facilitate a user
(from 3.1 User support for optimised processing)

3.1.3 depending on loading figures
(from 3.1 User support for optimised processing)

3.2.1 depending on loading figures
(from 3.2 System resources)

3.2.2 the system's scheduling must be run
(from 3.2 System resources)

3.3.1 the system shall use open source
(from 3.3 Compilers, software & licences)

3.3.2 the system shall provide compilers
(from 3.3 Compilers, software & licences)

4.1.1 the system shall provide dedicated
(from 4.1 VM creation and update)

4.1.2 the system shall provide dedicated
(from 4.1 VM creation and update)

4.1.3 the system shall enable a directory
(from 4.1 VM creation and update)

4.1.4 the system shall accept processes
(from 4.1 VM creation and update)

5.1.1 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.1.2 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.1.3 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.1.4 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.1.5 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.1.6 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.1.7 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.1.8 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.1.9 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.1.10 the system shall accept processes
(from 5.1 Testing; Standards for the process)

5.2.1 the system should facilitate a user
(from 5.2 Sharing common functionality)

5.2.2 the system shall record code modified
(from 5.2 Sharing common functionality)

5.3.1 the system shall use open source
(from 5.3 Compilers, software & licences)

5.3.2 the system shall provide compilers
(from 5.3 Compilers, software & licences)

6.1.1 the system must provide an input
(from 6.1 Functional display)

6.1.2 the system must provide an input
(from 6.1 Functional display)

6.1.3 the system must provide an input
(from 6.1 Functional display)

6.1.4 the system must provide an input
(from 6.1 Functional display)

6.2.1 the system shall provide a mesh
(from 6.2 Cataloguing)

6.2.2 the system shall provide a file
(from 6.2 Cataloguing)

6.2.3 the system's cataloguing sub-system
(from 6.2 Cataloguing)

6.3.1 the system should provide a list
(from 6.3 Data availability)

6.3.2 the system shall record provenance
(from 6.3 Data availability)

6.3.3 the system's cataloguing sub-system
(from 6.3 Data availability)

6.4.1 the system must provide sufficient processing
(from 6.4 Processing needs)

6.4.2 the system must provide processing
(from 6.4 Processing needs)

6.4.3 the system shall make the job run
(from 6.4 Processing needs)

6.5.1 depending on loading figures
(from 6.5 Processing needs)

6.5.2 depending on loading figures
(from 6.5 Processing needs)

6.5.3 depending on loading figures
(from 6.5 Processing needs)

7.1.1 carry out review of the system
(from 7.1 Create login and set up secure)

7.1.2 it is envisaged there will be
(from 7.1 Create login and set up secure)

7.1.3 it is envisaged there will be
(from 7.1 Create login and set up secure)

7.2.1 It is envisaged there will be
(from 7.2 Request resources or use common)

7.3.1 support for users and system
(from 7.3 Support for users and system)

7.3.2 support for users and system
(from 7.3 Support for users and system)

7.3.3 support for users and system
(from 7.3 Support for users and system)

7.3.4 support for users and system
(from 7.3 Support for users and system)

7.3.5 support for users and system
(from 7.3 Support for users and system)

7.3.6 support for users and system
(from 7.3 Support for users and system)

7.3.7 support for users and system
(from 7.3 Support for users and system)

7.3.8 support for users and system
(from 7.3 Support for users and system)

7.3.9 support for users and system
(from 7.3 Support for users and system)

7.3.10 support for users and system
(from 7.3 Support for users and system)

7.3.11 support for users and system
(from 7.3 Support for users and system)

7.3.12 support for users and system
(from 7.3 Support for users and system)
There are six components considered in the design which are described in the following section, and are related as shown in figure 2. The components have been developed alongside system use cases which are documented in later sections.

Figure 2: Component organisation

Components

Component : Archive Creation & Maintenance

Methods
Submit_data() : *Data from any source may be stored in the system: it must have adequate history/quality.*

Create_provenance_details() : *Acquire details of data’s source, processing history (incl s/w versions, dependencies and environment), any comments, etc. and attach as meta-data.*

Create_indexing_entry() : *Ensure the data may be found again by assigning sufficient unique identifiers to it. These should also be searchable (?SQL db).*
Store_data()  :  *Data is stored - the system will need to alert support personnel if disk space becomes limited. This step should include a check for errors in the product/metadata by support staff.*

**Contains class: ArchiveInputManager**

Class to oversee the addition of new data to the system. Will delegate:
- checking for pre-existing identical data
- creating an index entry for the data
- storing the data
- notifying the data producer (ingest or processing) of success/failure

Provides interfaces to receive a data record from the Data Ingest & Monitoring component, and to check for existence of a data file required by the Task Schedule component.

Note that much of the functionality may actually be manual at least to start with, and therefore fully automated ingest may only be an option for the future.

Will be strongly interdependent with the ArchiveRequestController class.

**Component : Archive Search & Extraction**

*Methods*

Search_data()  :  *Submit some criteria to the store of unique data identifiers to find a match, or selection of matches.*

Submit_data_request()  :  *Request a specific dataset.*

Check_for_data_dependencies()  :  *Given a list of required datasets, ensure all are present. It will not be up to the archive to suggest alternatives, the requester (processing chain) will either need to re-submit a list should one or more of the original set not be available, or notify the task scheduler that the job is suspended until the correct data is available. In either case, the scientists should be informed.*

**Contains class: ArchiveRequestController**

A controller to manage and delegate search and retrieval tasks within the component. Given a data file request, the correct data will be found and passed to the requester.

Will be strongly interdependent with the ArchiveInputManager class.

NOTE: Jasmin-CEMS already has a rudimentary archive search facility. It would be beneficial to incorporate and advance this if possible.

**Component : Data Ingest & Monitoring**

*Methods*

Check_for_data()  :  *A periodic query to see whether required data has arrived. Might be that this acts on the archive, or could be in the other direction i.e. an event raised by the data ingestion system to alert subscribers to new data acquisition.*
Contains class: **DataIngestManager**

Class to oversee collection and management of source data from various providers.

Will delegate:
- notifying dependent processes of data absence
- retrying data request from providers
- prompting task schedule controller to restart task when data becomes available
- perform common QA checks on input (and make info available to processing chains).

Will use interface on ArchiveInputManager class to store data.

**Component: Environment**

The environment component realizes many non-functional requirements.

It is also the interface for the processing chain software to use the system to reduce the former's dependencies on specific component implementations.

All operations listed here are manual processes, with the exception of automatically executing a job on a VM (instigated by the task scheduler), and swapping the test and live VMs over.

**Methods**

Log in and locate owned test VM() : `[manual]`

Create runnable s/w on test VM() : `[manual]`

Test s/w() : [manual]

This should be automated by the processing chain projects ideally.

Make tested s/w live() : [manual]

Set the process going for swapping VMs over.

Log in and locate preferred VM() : `[manual]`

Submit_job_request() : *Envisage this to be an interface/GUI to allow users to set processing in train.*

Log in to system() : [manual]

Review system statistics() : [manual]

Start_s/w_execution_on_VM() : *Operation used by the task scheduler to start a processing chain.*

Swap_VM_roles() : *When VMs change over from test to live, the task scheduler will need to know, and there may be implications for the monitoring and archiving activities as well.*
Figure 3: Contents of Environment component

**Contains: UserSupport**
Ensure usability and support for system users from processing chain through to data customers.

**Contains: Validation**
Defined process for ensuring that all s/w is to acceptable standard. Includes using common libraries.

**Contains class: AccreditationManager**
Responsible for gathering provenance information for each processing chain and for each data output.

**Contains class: VM_Management**
Oversight of VM administration:
- creation of new VM for each live processing chain
- VM test bed for each processing chain
- easy method to alternate test/live VM
- ‘spare’ VMs for people wishing to try a processing chain

**Component: Process Monitoring**

**Methods**
Check_capacity() : *With multiple cores available, it would be wise to have an automated way of distributing jobs - this may already be available.*

Retrieve_usage_statistics() : *For support personnel to see how the system is performing. Processing and archive/disk usage stats.*
Figure 4: Contents of Process Monitoring component

Contains class: CoreMonitor
Oversees load on CPUs and provides the peak/trough data for better scheduling of tasks.

Contains class: MemoryUseMonitor
Class to oversee the use of volatile and non-volatile memory. Will delegate to specialist class for each. Responsible for presenting the results to the user and issuing warnings when limits are reached; these limits will form part of the configuration settings pertinent to the role.

class: DiskUseMonitor

class: RAMUseMonitor

Component: ProcessingChainSW
This is a representative component for all the processing chains which will each run in a VM.

Methods
Run() : Set a processing chain going.

Get_list_of_required_data() : The dataset production s/w needs to be able to tell the system what datasets it needs for any particular run in order to manage automation/suspend/retry procedures.

Component: Task Schedule

Methods
Notify_data_aquisition() : When data is archived/stored, the scheduler should be informed in case any suspended jobs are waiting for that data.
Amend to new s/w version/location() : Need to know that VMs have swapped.

Check_availability() : Check that there is sufficient capacity to run a job.

Run_job() : Manually request that scheduler runs a job.

Retrieve_planned_system_tasks() : Review upcoming jobs in the scheduler.

Create_task_entry() : Set up task to run a processing chain job at a specific time or on receipt of specific dataset.

Task_initiated() : This might be more or less the same as Run_job(). Ensure processing chain submission fits into the current allocation process used by Jasmin-CEMS for access to the Lotus cluster (fair share algorithm). Note that priority access is given to funded users who therefore have a dedicated queue with priority access to processing.

Check_scheduled_tasks() : The scheduler will need to keep on top of its list of what to run and when.

*Contains class: ScheduleController*

Looks after all aspects of queuing and ensuring processing chains are run in a timely manner. Will delegate:

- notifying 'user' of task delay or failure
- retrying task (dependent on user/environment settings)

Will use interface on ArchiveInputManager class to check for required data files in a timely manner. This will depend on where in the processing chain the data is required: it may be possible to suspend tasks.
Use Case Analysis

Actors

DataConsumer
Role where the user searches for, locates and acquires a data product.

DataProvider
Role of supplying data to the archive. This may be raw source data from satellite download, or it may be products from an external processing chain or other source. This role requires access to dataset provenance when archiving data so that the information is maintained.

ProcessingChainProvider
Research or other group responsible for providing the processing chain software and maintaining its operation.

Scientist
General scientific user role which may use the system via a generic VM and have limited access to its facilities. In particular, this role is limited in its ability to request a processing chain task.

SupportStaff
Essential role to support users and maintain the system. Responsible for technical advice, system monitoring, process adherence etc. This role encompasses current data archive activities, and is planned to expand to carry out the improved indexing of source data prior to storage.

Use Cases
The diagrams illustrate the relationship between use cases and between use cases and actors. The use case description follow and sequence diagrams for many of them are at Annex B.
Figure 5: Primary system use cases
uc Provenance Use Cases

Figure 6: Provenance and history use cases
Use case details

Augment dataset history
Provide comment or feedback on the dataset or a specific part of the dataset. The existing information will need to be accessed for additions to be made, then the completed updates stored for future access.

Create data index entries
Done by support when data ingested, done by processing chain for data products. Will need to integrate with JASMIN/CEMS current search facility using their MOLES data model, and with the planned improvements in this area.

Generate dataset history
Dataset history must be of sufficient standard containing ISO19115-compliant information on the data themselves, additional fields where as appropriate for unique identification of the data, and for data
products, all system production history i.e. system environment, library versions, source data identifiers, processing chain component versions etc.

**Initiate ad-hoc run**
Processing chains will be available for scientists with login credentials to the Jasmin/CEMS system. TBD—should these be people with specific access to the dedicated VM, or should the processing chain be available as a task (with set parameters) from any valid VM? If the latter, this could be via the task scheduler which, if a web page, could allow validated users to access it and submit jobs. It is suggested that they take lower priority than scheduled jobs.

**Inspect dataset**
Simple review of data products to check for their presence and correctness.

**Monitor system**
Support staff will need to monitor system performance for processing and storage capacity. In any initial trial implementation, metrics should be gathered in order to plan future requirements.

**Retrieve data**
Accessing the data archive for required source data or product. This use case will be as a result of a successful search, and will include retrieval of the dataset's history detail.

**Retrieve dataset history**
Full data provenance must be available to users whenever the data itself is accessed. In addition, it should be viewable in advance to assist users in deciding whether they wish to retrieve the data. It may be an option to provide a selection of data histories corresponding to datasets which fulfil search criteria.

**Run processing chain**
Once started, the processing chain must be managed so that data oversight is possible. This is the interaction with the task scheduler notification of missing input data, and the subsequent choice by the scientists to alter the requirement or wait for the original request. On completion, the s/w must ensure that indexing and history information is created for the product and that it is submitted for upload into the archive.

**Schedule automatic run**
Via the task scheduler, processing chains can be set to run regularly by the owning research group scientists. The simplest approach would be to have this on a timer with the scientists alerted if source data isn't present, rather than the scheduler working on a data arrival event-driven algorithm.

**Search archive**
Enhancement of the data discovery catalogue in current use on JASMIN-CEMS to include indexing at file level (current ongoing work at CEDA). Need to ensure requirements match up here and/or additional work is fully integrated with the existing components.

**Set up VMs**
VMs should be set up on the managed system so that support staff maintain control of all updates and can ensure a configuration control process is followed. Upgrades to core services and libraries will need to be tested first in the test VM per research group. The aim is to have all groups using identical VMs and to share libraries wherever possible (e.g. file readers). All of a processing chain's dependencies are present on the VM.

**Swap test & live VMs**
When all tests pass on the backup/test VM, then it becomes 'live' and the original VM becomes the backup. The task scheduler will need to be informed of the change so that the processing chain job is run on the correct VM.

**Upload input data/product to archive**
Addition of data to the archive is as a result of source data becoming available, scientists offering datasets or a processing chain product being created. In all cases, indexing and provenance information should be checked by support staff (at least initially) before formal submission to the archive.

**Upload new version of processing chain software**
When an updated version of the processing chain s/w becomes available, it must run through its comprehensive set of tests on the backup VM. This checks that all processing chain's dependencies can be accessed, any environment upgrades pass test, while the 'live' VM is untouched.
### Annex A: Traceability matrices

**Relationship Matrix: Source** Primary Use Cases **Target** Requirements Model **Relationship:** Realization:Source -> Target

<table>
<thead>
<tr>
<th>Source Component</th>
<th>Target Requirement</th>
<th>Model Relationship: Realization</th>
<th>Source Requirement</th>
<th>Model Relationship: Realization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive Creation &amp; Maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Archive Search &amp; Browsing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Import &amp; Monitoring</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Monitoring</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Quality</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Schedule</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relationship Matrix: Source** Components **Target** Requirements Model **Relationship:** Realization:Source -> Target

<table>
<thead>
<tr>
<th>Source Component</th>
<th>Target Requirement</th>
<th>Model Relationship: Realization</th>
<th>Source Requirement</th>
<th>Model Relationship: Realization</th>
</tr>
</thead>
</table>
Annex B: Use case sequence diagrams
sd Monitor system

SupportStaff

Environment

Process Monitoring

Task Schedule

Log in to system()

Review system statistics()

Retrieve usage statistics()

Retrieve planned system tasks()
sd Run processing chain

Task Schedule

Environment

Archive Search & Extraction

Processing Chain SW

Upload input data/product to archive

Data Ingest & Monitoring

Check_scheduled_tasks()  
Task_initiated()  
Start_s/w_execution_on_VM()  
Get_list_of_required_data()  
Check_for_data_dependencies()  
all data present

{data}

Run()  
Submit_data_request(id)  
Data files (products)  
Upload data()  
Check_for_data()

{no data}

Suspend task()  
loop data monitoring

Check_for_data()
sd Schedule automatic run

ProcessingChainProvider

Environment

Task Schedule

Log in and locate preferred VM()

Create_task_entry()

(from Actors)
DataConsumer

Log in and locate preferred VM()

Search_data(criteria)

Data fulfilling criteria()

Submit_data_request(id)

Dataset()

(from Actors)
Upload data to archive

DataProvider

Archive Creation & Maintenance

Task Schedule

Submit_data()

Create_provenance_details()

Create_indexing_entry()

Store_data()

Notify_data_acquisition()

(from Actors)
Upload new version of processing chain software

ProcessingChainProvider

Environment  Task Schedule

Log in and locate owned test VM()
Create runnable s/w on test VM()
Test s/w()

alt Test results

[pass] Make tested s/w live()

[fail] do nothing

Swap_VM_roles()
Amend to new s/w version/location()

(from Actors)