

**Grant Agreement ECP-2007-DILI-527003**

**ARROW**

# **Specification of Rights Information Infrastructure**

## **V1.0**

|                                |   |
|--------------------------------|---|
| <b>Deliverable number/name</b> | <i>D5.2</i>   |
| <b>Dissemination level</b>     | <i>Public</i>   |
| <b>Delivery date</b>           | <i>12<sup>th</sup> October 2009</i>   |
| <b>Status</b>                  | <i>Final</i>  |
| <b>Author(s)</b>               | <i>Manfred Gravelius, Christian Kanja (MVB),<br/>Gabiella Scipione (CINECA)</i> |



|      |   |    |
|------|---|----|
| I.   | Executive Summary .....   | 3  |
| II.  | Arrow Basic Concepts.....                                       | 3  |
| 1.   | ARROW workflow.....   | 3  |
| 2.   | Actors .....  | 6  |
| 2.1. | Access users.....   | 6  |
| 2.2. | Metadata Providers .....  | 7  |
| 2.3. | System Management.....  | 7  |
| 2.4. | Service Monitoring .....  | 7  |
| III. | Detailed System Architecture.....                               | 7  |
| 1.   | ARROW Search and Retrieval Webservice.....                      | 9  |
| 2.   | ARROW Front End Services .....                                  | 10 |
| 2.1. | Interactive Web Portal.....                                     | 10 |
| 2.2. | Bulk Processing Web Portal.....                                 | 12 |
| 2.3. | Transformer Modules.....  | 13 |
| 2.4. | Direct API.....   | 13 |
| 3.   | ARROW Core Processing .....                                     | 13 |
| 4.   | ARROW Data Repositories .....                                   | 14 |
| 5.   | ARROW Connector .....   | 15 |
| 5.1. | Synchrone Connector .....                                       | 16 |
| 5.2. | Asynchrony Connector .....                                      | 16 |
| 5.3. | Special Connector.....  | 16 |
| 6.   | Orphan Works in a Nutshell .....                                | 17 |
| IV.  | Use Cases.....  | 19 |
| 1.   | Use case diagram for the librarians.....                        | 19 |
| 2.   | Use case diagram for ROW users (Rightholders, ROW manager)..... | 21 |
| 3.   | ARROW Workflow Sample .....                                     | 24 |
| 4.   | Sample Workflow Instance.....                                   | 25 |
|      | Annex I: Arrow System Design .....                              | 26 |

# ARROW - Specification of Rights Information Infrastructure

## I. Executive Summary

The present document contains the functional specifications of Arrow as derived by the analysis carried out in the WP5 (Design of ARROW system architecture). In technical terms, this document defines the system architecture for the management of rights information and orphan works registry. The explanations are based on the conceptual work done in WP4 and WP5 so far and are supposed to transform the results in a specification to allow the system set up. The system components are described with regard on the definitions of the expected workflows along the value chain in metadata provision which have been analysed and designed before.

In order to meet the requirements of performance, scalability, and flexibility, it has to be considered that the architecture should

- maintain the bibliographic rights metadata in a distributed network of resources
- allow new partners (as metadata providers) to join the ARROW federation with minimum efforts
- design a distributed search technique that takes into consideration a decentralized structure where external sources are linked together in a connected network

Standard methodology has been used for producing this document. While in section II “ARROW Basic Concepts” an overview about the principal ARROW workflow and the ARROW actors (users, providers, managers) is given, the design of the system architecture is described in section III as detailed as possible, considering the current status of the project. In section IV use cases are shown in order to explain some functionalities of the network system.

This report is provided by considering the facts and findings known as the status quo of the project. Possibly the present document will continue to evolve, by enriching the specifications, with the implementation of the Arrow system and with more detailed results gained from further investigations in the different work packages during the duration of the project.

## II. Arrow Basic Concepts

### 1. ARROW workflow

The purpose of ARROW is to support libraries wishing to digitize a book and use it. The Rights information Infrastructure will facilitate the process of carrying out a diligent search by means of a comprehensive system. This system will be able to provide the libraries with answers as to whether they are allowed to digitize a book or not.

The information needed to fulfill the purpose includes:

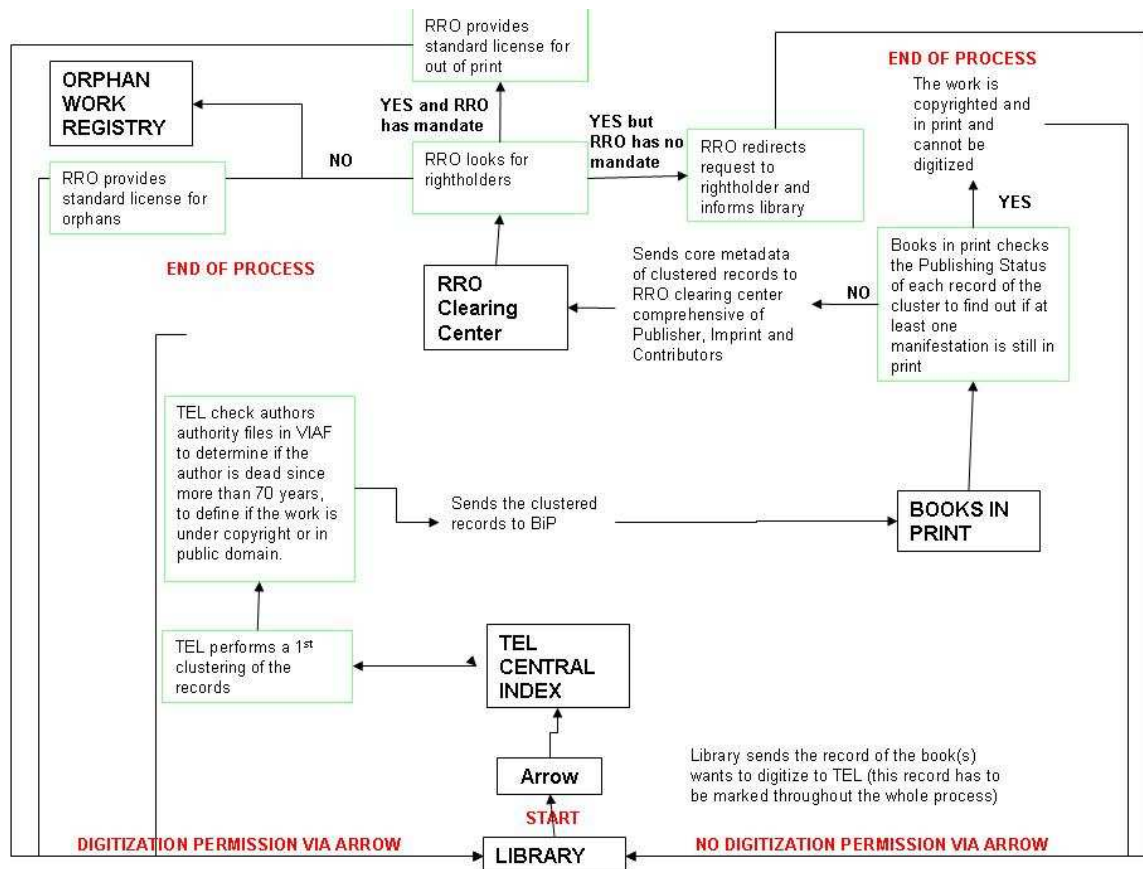
- the rights status of the work: if it is in public domain or copyrighted, if it is in print or out of print.

- the rightholders of the work: publishers, authors and other contributors, or their agents, that is organisations entitled to manage rights on behalf of rightholders.
- terms and conditions under which digitisation and use are allowed

Information needed is stored in the systems of different organizations for their own specific purposes. The key players identified so far are: The European Library, Books in Print databases, Reprographic Rights Organizations systems, Authority Files (VIAF)

The diagram below shows the basic system workflow of ARROW as an overview. It starts from a library as a potential user that wishes to digitise a book and shows the flow that the Arrow system has to build to provide a response.

Several actors concur to this definition; those identified so far represent the Metadata providers of the system. In the workflow below the providers that have been taken into account are the domain-specific ARROW partners (libraries as provider (TEL), BIP providers and RROs on a European level.



The finalisation of the workflow is still in progress and depends to a certain degree on given external elements as well as on further external developments (e.g. the provision of adequate clustering algorithms). Thus the workflow will still have to be adjusted and optimized during the development phase and according to the experiences reached in the pilot phase.

Description of the steps in the workflow:

**Step 1.** Library submits a query to ARROW

**Step 2.** ARROW “forwards” the query to TEL

**Step 3.** TEL processes the request, identifying the book by matching the metadata in the query with the records in the Central Index TEL

**Step 4.** TEL “forwards” the query to VIAF that adds further data such as Author(s)/Contributor(s) name and date of Death

**Step 5.** TEL performs a clustering process and “forwards” the cluster to ARROW

**Step 6.** ARROW processes the information and checks the copyrighted or public domain status of the cluster

**Step 7.** ARROW checks Country of Publication and puts the cluster on the correct processing line (on country basis)

**Step 8.** ARROW “forwards” the cluster to Books in print (or equivalent)

**Step 9.** BiP processes the information and adds for each record in the new cluster:

- Publication Date
- contact detail of publisher/imprint
- print/out of print status

**Step 10.** BiP “forwards” the cluster to ARROW

**Step 11.** ARROW processes the information and checks the in print/out of print status of the cluster

- if at least one manifestation in the cluster is in print, the cluster representing the work is to be marked as “in print”
- if all manifestation in the cluster are out of print, the cluster representing the work is to be marked as “probably out of print”
- if the work is in print ARROW provides feedback to the library in terms of
  - o in print status of the work
  - o publisher contact detail of latest editions
- if the work is “probably out of print” or in public domain ARROW redirects the request to the appropriate RRO
- ARROW stores the information for further reuse

**Step 12.** ARROW “forwards” the cluster to the RROs

**Step 13.** RRO processes the request, matches the cluster with its repertoire and search for rightholders (contributors and publishers) according to internal workflow for out of print and orphan works management:

- if rightholders can be traced or contacted, the work is marked as “out of print” and RRO carries on the process
- if rightholders can not be traced or contacted, the work is marked as “probably orphan”

**Step 14.** RRO declares the work as “probably orphan”

**Step 15.** RRO provides feedback to ARROW and ARROW to library

## **2. Actors**

In the present paragraph we identified the different actors authorised to use Arrow when this will become a widely used service. They are here grouped in three categories: access users, metadata providers and system managers.

### **2.1. Access users**

#### **End Users**

End users are human users that can perform search activities on the Arrow system through the Interactive web portal and Bulk Processing web portal of the front-end services. We identified two user profiles among them:

##### *Librarian*

This is the main actor of the Arrow system, he/she queries Arrow as part of a diligent search in order to collect the needed information to digitalise a book and make use of it for different purposes. They cannot access the Arrow service without being registered in the system.

##### *Rightholder*

This is the main actor of the Registry of Orphan work and can be the rightholder itself, an agent or an organisation like the RRO. He/she needs to have the possibility to search/browse in the ROW and possibly claim the rightsownership on an orphan work. The access to the ROW to browse the catalogue is public; to claim for rights he/she needs to provide contact data.

#### **External Applications**

These are applications belonging to organisations external to Arrow that prefer to submit queries in the Arrow services through external systems that connect to the Arrow web service, like the library IT services that submit queries in the Arrow services via B2B to the Arrow web service.

## 2.2. Metadata Providers

Being Arrow conceived as a federated system, there will be different organisations that will provide metadata and other information in order to define the status of a work/manifestation.

We identified two groups: Metadata Providers and Metadata Aggregators. Metadata Providers are all the single organisations identified in the deliverable 5.1 within the Library, Books In Print and RRO domains. Metadata Aggregators are those organisations that act as aggregators of metadata, such as The European Library, Europeana, VIAF (The Virtual International Authority File), etc...

## 2.3. System Management

### System Configuration Manager

These are system administrators that are entitled to define the different user groups that can access the Arrow service in order to guarantee the access to the different profiles of services to all the actors mentioned in this paragraph.

### System Administrator

These are people responsible of the maintenance of the Arrow system, taking care of the ordinary maintenance of the Arrow system.

### ROW Manager

These are people belonging to the different organisations and in charge for the management of the Registry of Orphan Works at national level.

## 2.4. Service Monitoring

These are people that provide feedback to the final users when they need some help to access the Arrow service. Being the requests coming from final users of different nature, we defined two levels of help desk:

- Editorial Help Desk
- Technical Help Desk

In this category of services we identified also the role of a person entitled to analyse the results of the different clustering processes in the Arrow workflow in order to verify their correctness.

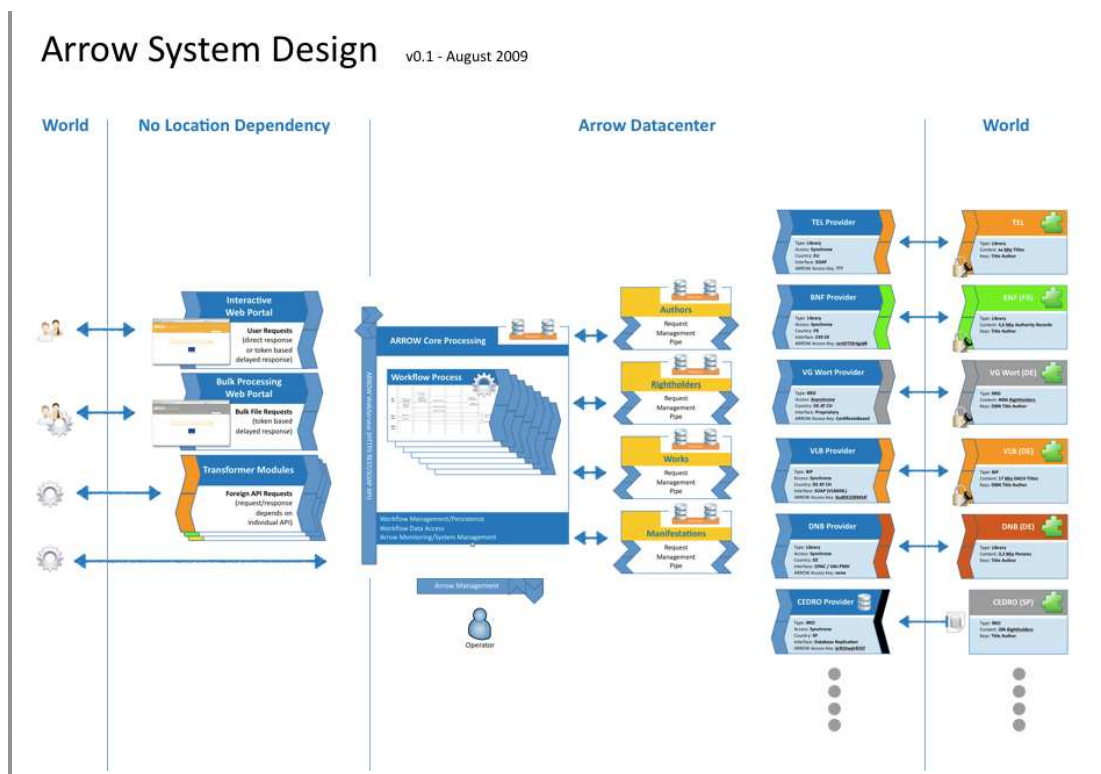
- Clustering and Results Validator

# III. Detailed System Architecture

Based on the information about the requirements the ARROW system architecture must fulfil there is a strong demand to break the complexity into specialised modules. These modules are much easier to develop, easier to maintain, similar tasks can be adapted into similar modules, the modules itself

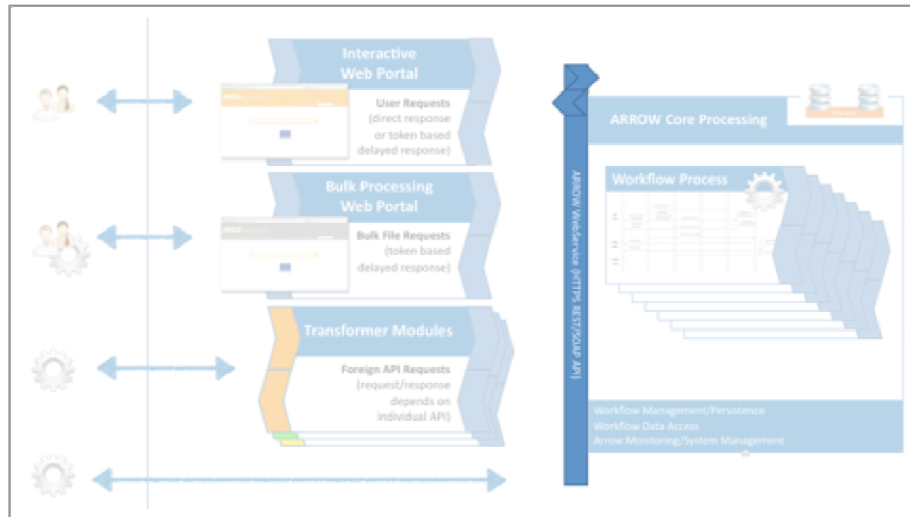
are well defined and prepared for unit testing. And the overall scalability is not depending on a single high performance system but can be achieved by using horizontal scalability and distributed module deployment.

In addition there is currently not foreseeable that all kind of requesters and all kind of data providers are able or interested in developing a standard B2B interface for the ARROW project. That said, there should be flexibility in connecting to all kind of data repositories and offers all kind of search connectivity to make the ARROW project successful in an acceptable time frame. To achieve this flexibility, we decided to offer an open front end, able to be transformed in all kind of existing request methods, and a provider based backend, which encapsulates the individual access and retrieval methods of library, RRO and BiP repositories and offers these in a standardised way for the workflow itself.





## 1. ARROW Search and Retrieval Web Service



The primary access to ARROW is implemented as a REST or/and SOAP Web Service. This Web Service is accepting synchronous and asynchronous search requests and answers these requests by offering the response record or a token for asynchronous record lookup. It needs the input of various project members to define this interface because it needs to offer the best possible feature set based on the expected protocols used to retrieve ARROW data. It may even inherit it's fundamental design by using a given protocol like SRU / CQL and extending this protocol with the special needs of the ARROW data retrieval.

A schematic sample implementation of a typical single item ARROW search could look like this:

```
{
  "Title"      : "Das Kapital: Kritik der politischen Oekonomie.",
  "Language"   : "DE",
  "Authors"    : [
    { "Name"    : "Marx, Karl" },
    { "Name"    : "Engels, Friedrich" }
  ],
  "Publisher"  : "Verlag von Otto Meissner",
  "Published"  : 1867,
  "Country"    : "DE",
  "Pages"      : 903,
  "ARROW Query Type" : "Sync",
  "ARROW Query Version" : "1.0"
}
```

The interface itself should be a public interface with a documented API to be implemented in automated retrieval systems. For common access methods like interactive searches and bulk file uploads the interface is not directly used but instead a front end system is used to offer the interface, user management, bulk data processing, etc. The front end systems are individual

development projects and the only dependencies with the ARROW core is the well defined Web Service API.

## 2. ARROW Front End Services

### 2.1. Interactive Web Portal



The simplest public interface to ARROW will be a search engine like interactive web portal interface. It can be used for all kind of search requests but it is always only a single item processing. The following searches are supported:

**Synchronous Search:** A given search record (title, author, etc.) will be searched in ARROW. The ARROW core processing will use its internal repositories and all synchronous accessible external repositories. The search result will interactively presented to the requestor.

It is important to understand that the synchronous search may be announced on a later stage of the project because in the beginning there is not enough data available to make the synchronous search very valuable and very few synchronous external repositories are available. But the concept of the ARROW system architecture provides the strength of local repositories and fast, synchronous connection to valuable data partners. That said it may become very soon an important feature to have the opportunity to ask ARROW for a fast response when researching rights, especially if the ARROW repository has learned thousands of matched rights to give fast and reliable answers for common requests.

**Asynchronous Search:** A given search record (title, author, etc.) will be searched in ARROW. The ARROW core processing will use its internal repositories and all synchronous and asynchronous accessible external repositories. The requestor will not receive an interactive response but instead receive a token to request the result at a later time. In addition the front end could implement email notification or RSS feed for pushing the result to the requestor.

**Fast Search/Known Orphan Works Search:** A given search record (title, author, etc.) will be searched in ARROW. The ARROW core processing will **only** use it's internal repositories. The search result will interactively presented to the requestor.

The interface of the Interactive Web Portal could look like this:

The screenshot shows a web browser window titled "ARROW Web Portal" with the address bar containing "http://search.arrow.eu". The main heading is "ARROW Interactive Search Portal". Below the heading are three tabs: "Search", "Advanced Search" (which is highlighted), and "Orphan Works Search". The search form includes the following fields and options:

- Title (Manifestation):
- Title (Original - if known):
- Authors (semicolon separated):
- ISBN/ISSN (if available):
- Publisher:
- Country of Publishing:
- Language:
- Published:
- Pages:
- ARROW Search Type:  Interactive Search  Long Running Search
- Search button:

When searching a long running query (asynchronous search) the response may take several days. To make these kinds of searches usable there needs to be a status interface by web/feed/mail:

The screenshot shows the same browser window, but the search form is replaced by a status message:

Long Running Search "Das Kapital..." is queued.

Please visit your personal response page for the status of your search request:  
<http://search.arrow.eu/response?25dd10f3-9dd0-4d8c-a4b9-095f69687ab5>

You can follow your search status by [RSS Feed](#) or we can send you an email when your search is finished.

Your E-Mail:

## 2.2. Bulk Processing Web Portal



In addition to a simple single item search web portal the bulk processing web portal offers an upload interface for retrieval lists. The type of implemented lists needs to be defined, most common lists could be CSV or MARC-based sets from libraries.

The lists will be processed on the front end system and each list item will be transformed into an ARROW search record and trigger an asynchronous search request from the front end system to the ARROW web service. The whole management of the list is outsourced to the front end system and not implemented into ARROW. The front end system manages the retrieved ARROW job tokens and enriches the result set of the list whenever there is new data for the requested items available. The requestor can use the front end portal to see the status of the uploaded list items and may be informed by email or RSS about status changes.

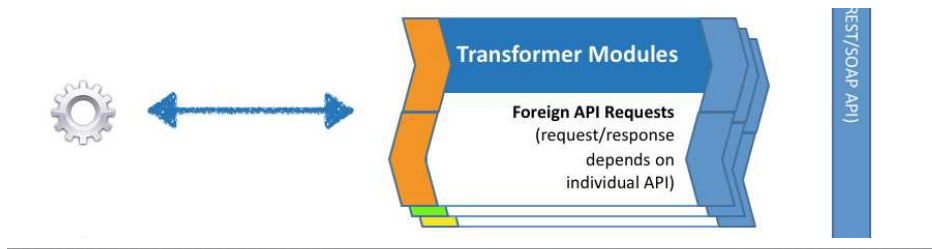
When finished, the result set can be downloaded from the front end system.

To get a better understanding about what the bulk processing could look like, here's an example:

*A library uploads a CSV with 10 books. The bulk front end parses the 10 lines and internally it sends 10 (async) separate requests to the ARROW Web Service. In the ARROW answer packets the bulk front end receives 10 tokens for the queued requests.*

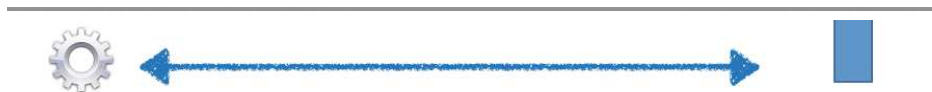
*The bulk front end is now pulling the Web Service for changes in the individual requests. It may happen that after some seconds 2 items are known as orphan works, after some minutes 2 others are known as still in print. These "news" are used to "enrich" the result set, meaning that these responses are stored in the bulk front end and depending on the settings these "news" may be sent by email or offered as an RSS feed to the requestor. Or the "enriched" CSV - meaning there are now 4 of the 10 items with a final result - is available for download as an intermediate result.*

### 2.3. Transformer Modules



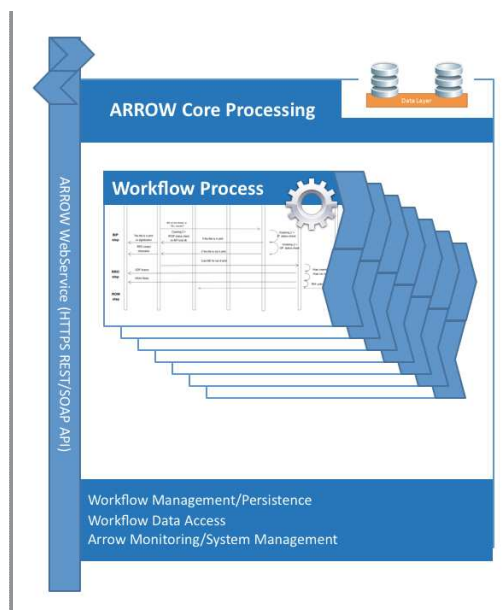
Depending on the final definition of the ARROW Web Service API it may be helpful to develop transformation front end systems to translate special widely used search protocols to the ARROW protocol. This can be done without changes into the ARROW core system but instead creating a transformation front-end system which interacts with the native ARROW protocol.

### 2.4. Direct API



When ARROW is an established and widely used service it makes sense to implement ARROW searched into existing or new services. This can be done by directly interacting with the well defined ARROW search and retrieval Web Service protocol.

## 3. ARROW Core Processing



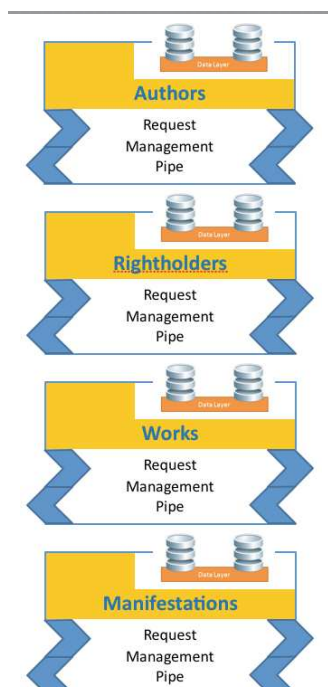
The ARROW core processing system is a database driven workflow engine. All requests received by the Web Service create a search record in the core database and depending on the type of request (synchronous, asynchronous, fast - additional types may be defined in the future) a highly optimised workflow process specialised for the kind of request will be instantiated. This process is hosted and managed in the core processing module but may be distributed to any amount of parallel running processing hosts (horizontal scalability). The workflow instances are persisted in the core database and are able to idle (sleep/wakeup) when waiting for asynchronous responses from the repositories. In contrast the synchronous requests are triggering synchronous workflow instances - the data is also persisted but the job itself is only asking for synchronous available repository data and is running from start to end of a request.

The isolation of the workflow itself offers the flexibility to build a future-safe and reliable ARROW system. When new repository types are available or the decision strategy changes over time the whole change management process depends on changed workflow blueprints - and these blueprints (definition files) can be added into the running ARROW system.

A sample of this process is shown in section IV. (Use Cases).

In addition the core processing system needs to offer an internal monitoring/management interface to operate the ARROW system.

#### 4. ARROW Data Repositories



The ARROW workflow processes do not know anything about external repositories or data sources. The workflows work with the internal ARROW data repositories to fulfil the retrieval task. These internal data repositories are database supported specialised retrieval modules which work with the

ARROW data providers to implement the best possible data aggregation strategy for a given search type and remember the results to optimise future requests.

In detail the workflow requests a data repository for a given type of search item and enriches the request with the already available metadata (search cluster). The data repository will first lookup it's own database for a possible existing answer. In addition - depending on the search type, the internal result and the metadata - the data repository module will request the search on all available ARROW data providers which are registered to provide responses for the given search. That means i.e. that a synchronous request will not be given to asynchronous-only data providers and that only the country, time, type and with time even more specialised repositories are selected.

For example a VLB data provider "knows", that VLB is able to offer fast responses (synchron) and the data is about German literature. If an ARROW request about a Spanish book is in the workflow, the ARROW repositories will search for data providers offering information about Spanish books - that means "by convention" the VLB will not be asked.

The results of the requests will always cached in the internal data repository. This could be enhanced by using a TTL (time to live) for the item to ensure fast responses but further validation especially for not conclusively identified items.

Again, as an example one of the "asynchronous" connected BiPs responds with the information "is still in print" about a book. Instead of asking the BiP again when the request for this book is repeated, the ARROW manifestation repository is caching this information. But this makes no sense "for years", because sooner or later the book may not be in print anymore. In this case a TTL of 30 days could make sense to have an adequate refresh interval for this kind of information, but there may be even faster TTLs if for example we integrate repositories which are in an early stage and currently build up their content.

Conclusively identified results will build the fundament for the orphan works registry - these items are finalised in the ARROW works-repository and could be retrieved with a special flag set in the ARROW Web Service request. The base for the definition of the status is always part of the data repository itself - it is persisted in the request history.

## 5. ARROW Connector

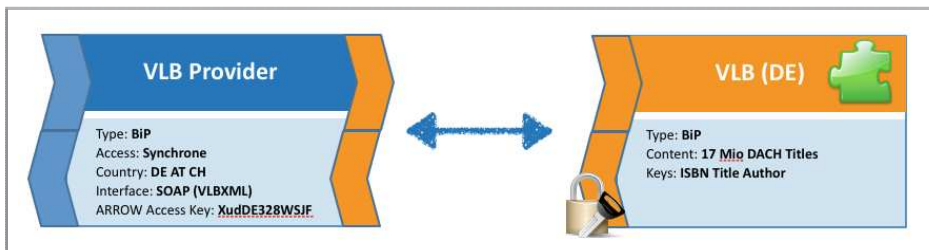
The external repositories (TEL, libraries, RROs, BiPs) offer an uncountable amount of different interfaces and protocols, different access security and some do even not publish any kind of external access to their databases (see D5.1). In the conclusion it is not realistic to base the ARROW system on the idea of a generic B2B interface homogenous available in all communications with these repositories.

Instead it makes sense to use a generic internal protocol between ARROW and a set of "connectors" which encapsulated the individual protocols and details of a connected repository. The connector "knows" about the access methods, security identifiers, interface types and search dialect in the external communication and translates it into the ARROW internal protocol.



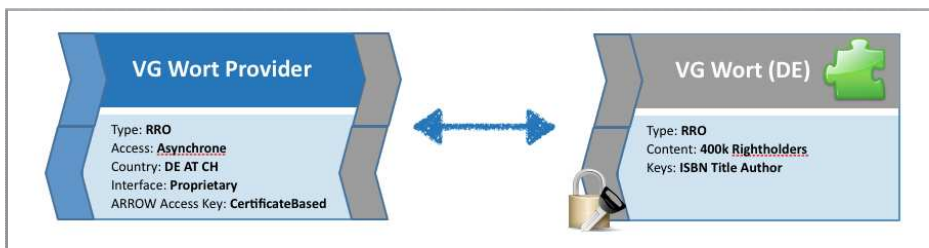
This means for the ARROW data repository it is able to request data from any kind of data source, especially data sources not even known today, without any knowledge, reconfiguration or coding.

### 5.1. Sychrone Connector



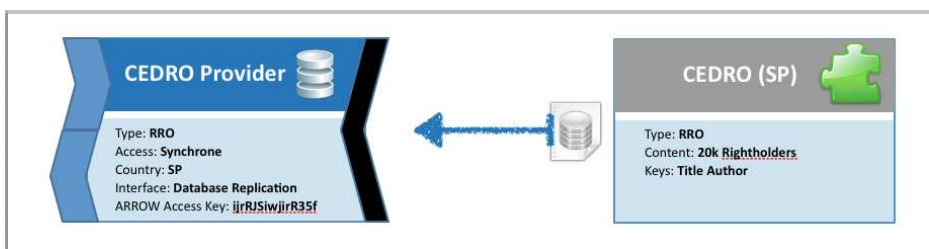
A synchronone connector connects with SOAP, REST or any other synchronone protocol to the external repository and receives the result within the same session.

### 5.2. Asynchrony Connector



An asynchrony connector manages the individual request token system of an external repository, which needs to lookup for the result through time consuming, maybe manual steps.

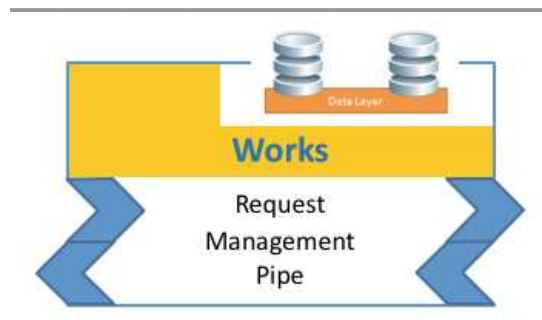
### 5.3. Special Connector



In addition to the “transformation” connector there may be “replication” connectors that receives database deltas or download snapshots from not interface-driven external repositories and provide these data internally like synchronous available data sets.



## 6. Orphan Works in a Nutshell

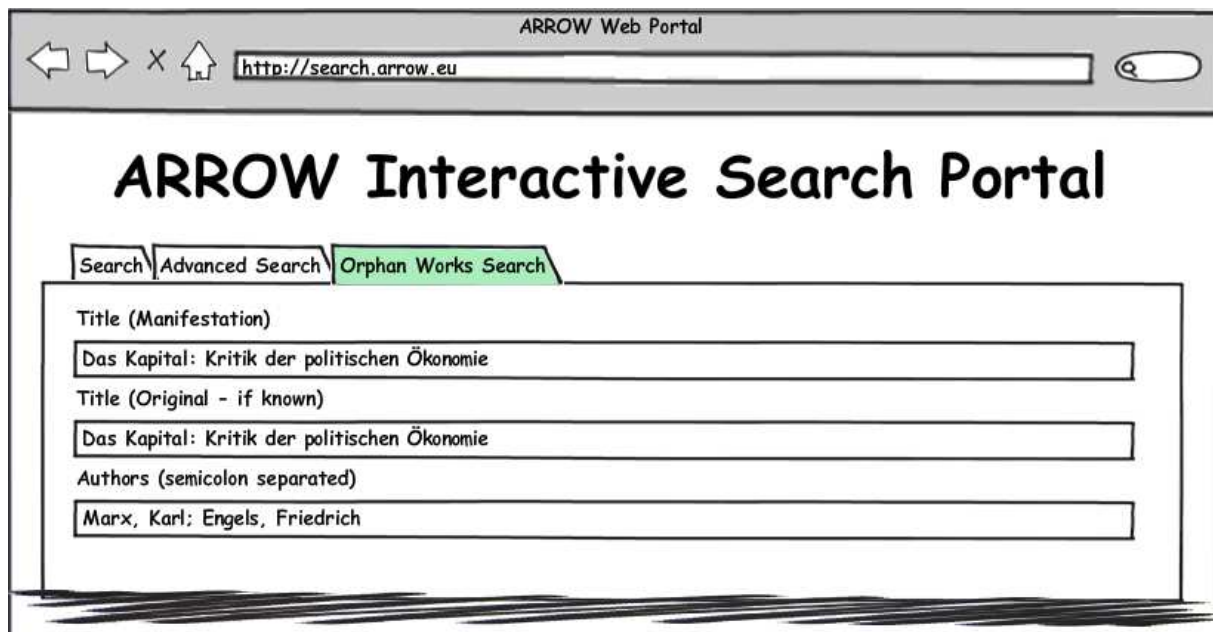


The ARROW system architecture is designed to offer an orphan works repository by implementing a “learning” works module. This module, which is one of the ARROW data repositories, stores all works found in the external provider databases during the day by day ARROW request processing and enriches this information with the calculated workflow decision and the trace back of “how” the decision was made and “where” the information came from.

The ARROW Web Service could implement a special search-type to search ONLY in the ARROW orphan works repository:

```
{  
  "Title"      : "Das Kapital: Kritik der politischen Oekonomie.",  
  ...  
  "ARROW Query Type"      : "Orphan",  
  "ARROW Query Version"  : "1.0"  
}
```

The core-processing module would trigger an equivalent “orphan only”-workflow, which would only search in the works repository for the given dataset. This Web Service request for orphan works is also part of the standard user interface described in section III.2.1.



ARROW Web Portal

http://search.arrow.eu

## ARROW Interactive Search Portal

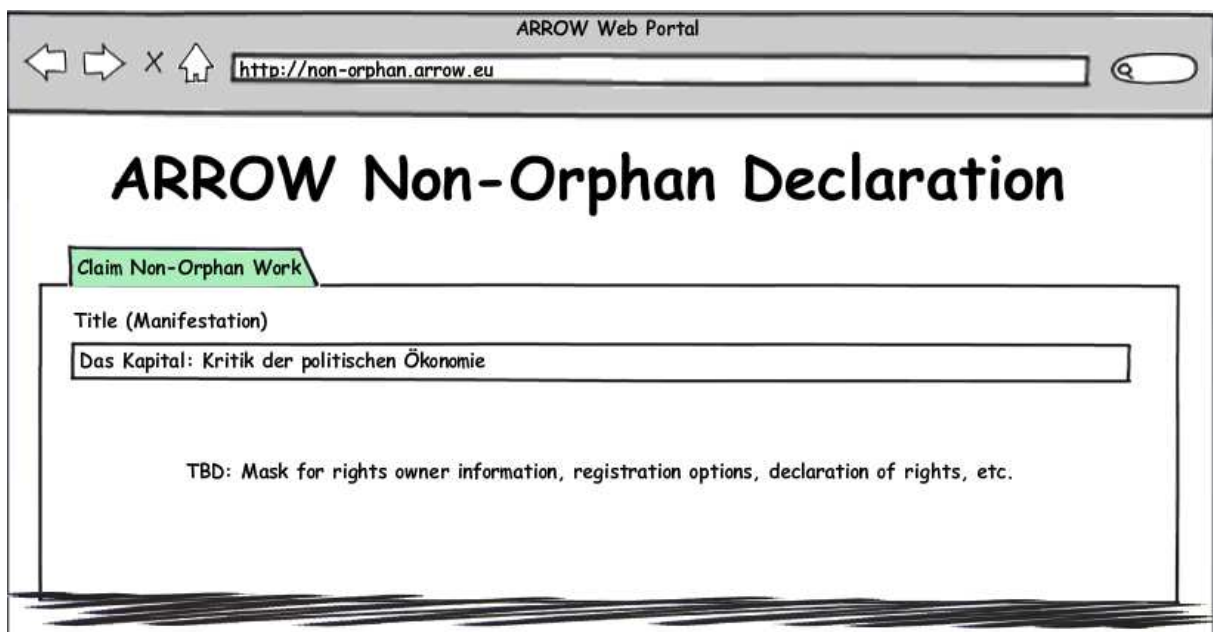
Search | Advanced Search | Orphan Works Search

Title (Manifestation)  
Das Kapital: Kritik der politischen Ökonomie

Title (Original - if known)  
Das Kapital: Kritik der politischen Ökonomie

Authors (semicolon separated)  
Marx, Karl; Engels, Friedrich

In addition to the automated learning of orphan works, there needs to be an interface for rightholders to declare his/her rights to wrong orphan declared items. The general process would be an orphan works rightholders front-end system, which offers an acceptable declaration and authoritative acceptance process. This front-end system uses the ARROW Web Service, which needs to have a special command type for “Non-Orphan Declaration” and an equivalent workflow to flag the works repository item with the new authoritative information.



ARROW Web Portal

http://non-orphan.arrow.eu

## ARROW Non-Orphan Declaration

Claim Non-Orphan Work

Title (Manifestation)  
Das Kapital: Kritik der politischen Ökonomie

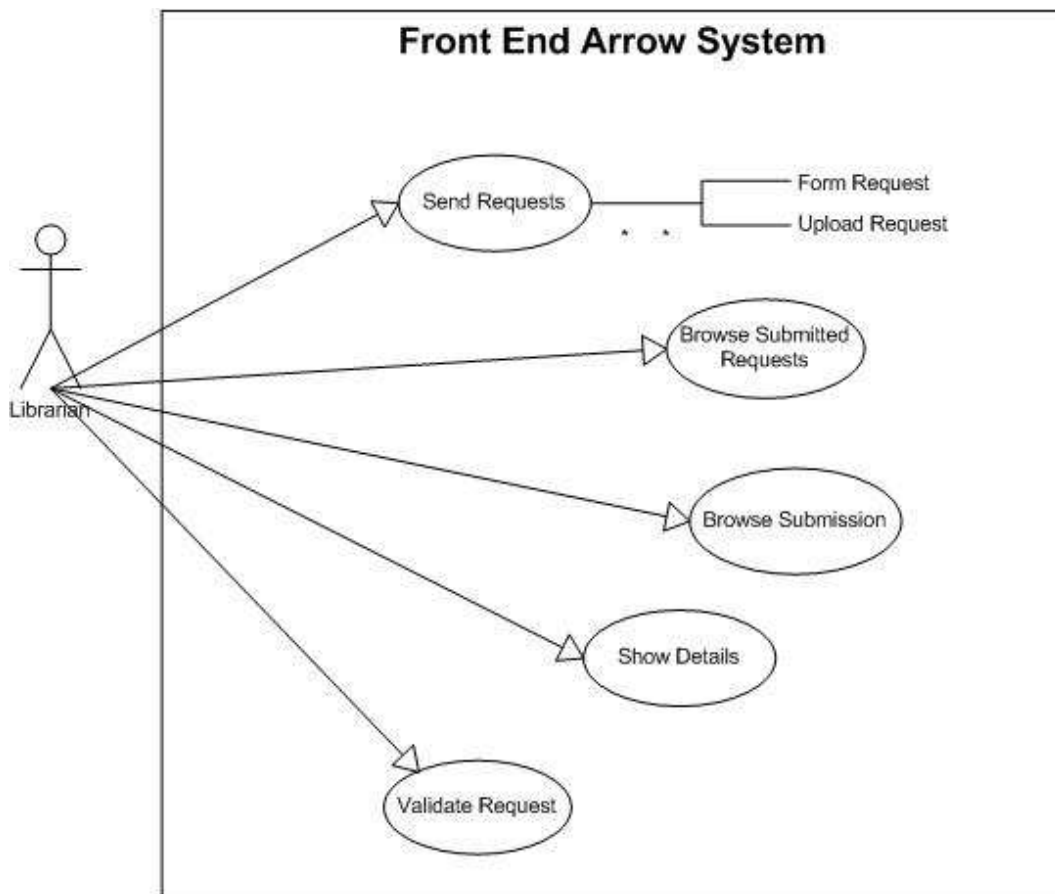
TBD: Mask for rights owner information, registration options, declaration of rights, etc.

This workflow and the detailed information what kind of documentation is needed to declare a non-orphan work needs to be defined in the responsible teams.

## IV. Use Cases

### 1. Use case diagram for the librarians

The aim of the present paragraph is to show what functions are foreseen at this stage in the Arrow front-end system for an actor like a library. This can be accomplished by using the use case diagram in the Unified Modelling Language (UML)<sup>1</sup> that presents a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.



Use case diagram for a librarian

From the use-case analysis we defined five functionalities/use cases, which are explained below in detail.

| <b>Use Case: Send Requests</b>  |
|---|
| <b>ID: UC1</b>  |
| <b>Actors:</b><br>Librarian   |
| <b>Preconditions:</b><br>The librarian has been authenticated by the system |

<sup>1</sup> The Unified Modeling Language - UML - is the most-used specification in the field of software engineering.. <http://www.uml.org/>

|  |
|--|
| <p><b>Events sequence:</b><br/>The use case starts when the librarian selects the functionality “send a request”</p> <ol style="list-style-type: none"> <li>1. The librarian submits the book record plus the specifications of what rights are requested to the ARROW system.</li> <li>2. The librarian can choose to submit a single request via the form web portal or multiple requests (a list of book records) via the Bulk Processing web portal.</li> <li>3. ARROW validates the query</li> <li>4. ARROW provides feedback to the library on validation result</li> <li>5. ARROW transforms the query received by the library in the format needed to query TEL</li> <li>6. ARROW submits the output of the transformation to TEL</li> </ol> |
| <p><b>Post conditions:</b></p>   |

| <b>Use Case: Browse Submitted Requests</b>   |
|--|
| <p><b>ID: UC2</b></p>  |
| <p><b>Actors:</b><br/>Librarian</p>  |
| <p><b>Preconditions:</b><br/>The librarian has been authenticated by the system</p>  |
| <p><b>Events sequence:</b><br/>The use case starts when the librarian selects the functionality “submitted requests”.<br/>The system shows the history of all the requests submitted by the librarian distinguishing between those pending, those who require a validation by the librarian and the closed ones.</p> |
| <p><b>Post conditions:</b></p>   |

| <b>Use Case: Browse Submission</b>  |
|---|
| <p><b>ID: UC3</b></p>   |
| <p><b>Actors:</b><br/>Librarian</p>   |
| <p><b>Preconditions:</b><br/>The librarian has been authenticated by the system.</p>  |
| <p><b>Events sequence:</b><br/>The use case starts when the librarian selects one of the submissions shown by the use case US2.<br/>The system shows the list of single requests (in the case of multiple queries) distinguishing between those pending, those who require a validation by the librarian and the closed ones.</p> |
| <p><b>Post conditions:</b></p>  |

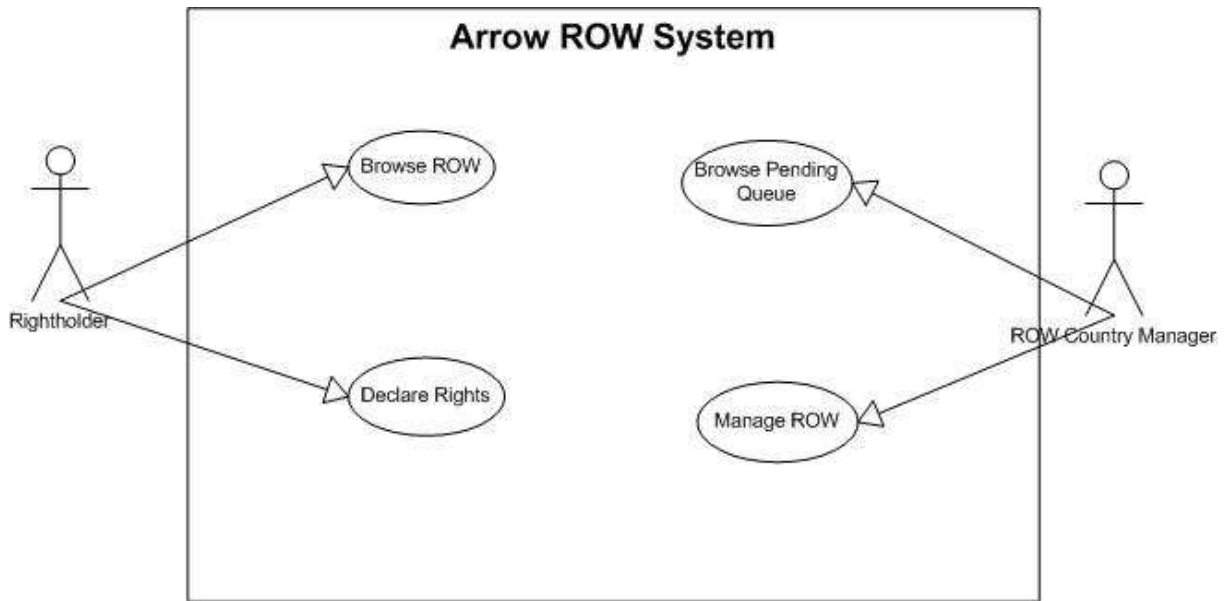
| <b>Use Case: Show details</b>       |
|-------------------------------------|
| <p><b>ID: UC4</b></p>               |
| <p><b>Actors:</b><br/>Librarian</p> |

|  |
|--|
| <p><b>Preconditions:</b><br/>The librarian has been authenticated by the system.</p>   |
| <p><b>Events sequence:</b><br/>The use case starts when the librarian selects one of the requests in the use case US3.</p> <ol style="list-style-type: none"> <li>1. The system shows the status of the request.</li> <li>2. If the Arrow processing terminated the system provides also the result of the query.</li> </ol> |
| <p><b>Post conditions:</b></p>   |

| <b>Use Case: Validate Request</b>  |
|--|
| <p><b>ID: UC5</b></p>  |
| <p><b>Actors:</b><br/>Librarian</p>  |
| <p><b>Preconditions:</b><br/>The librarian has been authenticated by the system.<br/>The matching of the metadata in the librarian query with the records in the TEL Central Index or in the Arrow Data repository provides a partial match. In this case the Arrow service requests to the librarian to refine the search and reiterate the query</p>   |
| <p><b>Events sequence:</b></p> <ol style="list-style-type: none"> <li>1. In the details of the arrow answer (US4) the librarian sees if it is necessary a refinement/validation</li> <li>2. The librarian selects among the different book records found by the system which is the book he is asking for information</li> <li>3. The system reiterates the query with the identified record.</li> </ol> |
| <p><b>Post conditions:</b></p>   |

## 2. Use case diagram for ROW users (Rightholders, ROW manager)

We foresee two kinds of Actors in the registry of Orphan Work: the rightholder and the ROW manager. The first has the possibility to search and browse in the ROW and possibly claims for rightsownership; the second is responsible to manage the registry and is entitled to assess the pending claims of the rightholders.



| <b>Use Case: Browse ROW</b>   |
|---|
|   |
| <b>ID: UC6</b>  |
| <b>Actors:</b><br>Rightholder   |
| <b>Preconditions:</b>   |
| <b>Events sequence:</b><br>The use case starts when the Rightholder selects the functionality “Orphan works search engine” <ol style="list-style-type: none"> <li>1. It will be possible to search an orphan work following different search criteria (to be defined)</li> <li>2. The system processes the request and shows the results</li> </ol> |
| <b>Post conditions:</b>   |

| <b>Use Case: Declare Rights</b>   |
|---|
|   |
| <b>ID: UC7</b>  |
| <b>Actors:</b><br>Rightholder   |
| <b>Preconditions:</b>   |
| <b>Events sequence:</b> <ol style="list-style-type: none"> <li>1. The Rightholder chooses the OW and claims for the ownership providing his contact data.</li> <li>2. The Arrow system validates the request and provides an acknowledge message to the Rightholder.</li> </ol> |
| <b>Postconditions:</b>  |

| <b>Use Case: Browse pending queue</b>  |
|--|
| <b>ID: UC8</b>   |
| <b>Actors:</b><br>ROW manager  |
| <b>Preconditions:</b><br>The ROW manager has been authenticated by the system.   |
| <b>Events sequence:</b><br>The use case starts when the ROW manager enters in the back office of the ROW and selects the functionality "Pending claims".<br>The system shows the history of all the claims submitted by the Rightholders for OW in which the country of publications corresponds to national area of the organisation (RRO, etc...). |
| <b>Post conditions:</b>  |

| <b>Use Case: Manage ROW</b>  |
|--|
| <b>ID: UC9</b>   |
| <b>Actors:</b><br>ROW manager  |
| <b>Preconditions:</b><br>The ROW manager has been authenticated by the system.   |
| <b>Events sequence:</b><br>The use case starts when the ROW manager enters in the back office of the ROW and see the status of the OW.<br>He has the permissions to modify the status of the OW, to cancel an OW and to insert a new OW. |
| <b>Post conditions:</b>  |

### 3. ARROW Workflow Sample

#### **Step 1:** Library submits query to ARROW

Create a library account. The account management is not part of this paper. It could be done partly in the front end systems (i.e. for personal user accounts in the single search interactive portal) or/and as part of the core ARROW system, based on given protocol authentication, federation with TEL or an ARROW internal account and subscription system.

The library would send a list of book records to the bulk front end system (1) or would ask for a single item search on the interactive web portal (2).

The ARROW system gets the request on an item by item query on the ARROW Webservice.

#### **Step 2-5:** ARROW “forwards” the query to TEL

(If this is defined in the workflow)

The workflow parses the request and queries the internal WORKS repository. The WORKS repository queries the TEL connector which knows the TEL search and retrieval language, the protocol specifications, the interface URI of TEL and about the ARROW/TEL-authentication process.

The TEL response is stored in the ARROW WORKS repository and returned to the workflow process.

#### **Step 6-10:** ARROW “forwards” the cluster to BiPs

(If this is defined in the workflow)

The workflow parses the request and queries the internal MANIFESTATIONS repository. The MANIFESTATIONS repository decides which data providers matches the query data and are able to enrich the cluster. The MANIFESTATIONS repository module queries the BiP data providers which knows the different BiP search and retrieval languages, the protocol specifications, the interface URIs and about the authentication process.

The BiP response is stored in the ARROW MANIFESTATIONS repository and returned to the workflow process.

#### **Step 11-14:** ARROW “forwards” the cluster to RROs

(If this is defined in the workflow)

The workflow parses the request and queries the internal RIGHTHOLDERS repository. The RIGHTHOLDERS repository decides which data providers matches the query data and are able to enrich the cluster. The RIGHTHOLDERS repository module queries the RRO data providers which knows the different RRO search and retrieval languages, the protocol specifications, the interface URIs and about the authentication process.



The RRO response is stored in the ARROW RIGHTHOLDERS repository and returned to the workflow process.

Step 15: ARROW provides feedback to the library

When the workflow is finished and all available information is queried, the workflow instance is set to finalised and the response packet is ready to be pulled from the front end system to integrate the data into the list (1) or to present the data to the interactive requestor (2).

#### 4. Sample Workflow Instance

This is a meta-code for a possible implementation of the general ARROW workflow

```
Record = Webservice.CurrentQuery
CALL ARROW.Works(Record)
IF Record.found
    CALL PREPARE_RESPONSE(Record)
END
CALL ARROW.Authors(Record)
CALL ARROW.Manifestations(Record)
IF Record.manifestations
    FOR EACH Record.manifestation
        CALL ARROW.Authors(Record.manifestation)
    END
CALL VALIDATE(Record)
IF Record.found
    CALL PREPARE_RESPONSE(Record)
END
CALL ARROW.Rightholders(Record)
CALL VALIDATE(Record)
IF Record.found
    CALL ARROW.Works(Record)           // store the result
CALL PREPARE_RESPONSE(Record)
END
```

Note: This is just a quick sample - there should be much more detailed investigation for an optimal search and valuate strategy. But this sample reflects the general isolation of the ongoing optimisation into a workflow model and the flexibility offered with such a platform decision.

# Annex I: Arrow System Design

## Arrow System Design v1.0 - August 2009

