D3.2v2: Design, implementation and deployment of Workflow Evolution, Sharing and Collaboration components

Deliverable Co-ordinator: Rafael González-Cabero

Deliverable Co-ordinating Institution: Universidad Politécnica de Madrid (UPM)

Other Authors: Raúl Palma (PSNC), Jose Gómez-Pérez, Aleix Garrido (iSOCO)

This document describes the M32 final release of the Wf4Ever WP3 components and specifications that are centred on the maximization of sharing and reuse of the preserved workflow-centric Research Objects.
**Wf4Ever Consortium**

This document is a part of the Wf4Ever research project funded by the IST Programme of the Commission of the European Communities by the grant number FP7-ICT-2007-6 270192. The following partners are involved in the project:

<table>
<thead>
<tr>
<th>Intelligent Software Components S.A.</th>
<th>University of Manchester</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edificio Testa</td>
<td>Department of Computer Science,</td>
</tr>
<tr>
<td>Avda. del Partenón 16-18, 1º, 7ª</td>
<td>University of Oxford, Oxford Road</td>
</tr>
<tr>
<td>Campo de las Naciones, 28042 Madrid</td>
<td>Manchester, M13 9PL</td>
</tr>
<tr>
<td>Spain</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>Contact person: Dr. Jose Manuel Gómez-Pérez</td>
<td>Contact person: Professor Carole Goble</td>
</tr>
<tr>
<td>E-mail address: <a href="mailto:jmgomez@isoco.com">jmgomez@isoco.com</a></td>
<td>E-mail address: <a href="mailto:carole.goble@manchester.ac.uk">carole.goble@manchester.ac.uk</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Universidad Politécnica de Madrid</th>
<th>University of Oxford</th>
</tr>
</thead>
<tbody>
<tr>
<td>Departamento de Inteligencia Artificial</td>
<td>Department of Zoology</td>
</tr>
<tr>
<td>Facultad de Informática, UPM</td>
<td>University of Oxford</td>
</tr>
<tr>
<td>28660 Boadilla del Monte, Madrid</td>
<td>South Parks Road, Oxford OX1 3PS</td>
</tr>
<tr>
<td>Spain</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>Contact person: Dr. Oscar Corcho</td>
<td>Contact person: Dr. Jun Zhao / Professor David De Roure</td>
</tr>
<tr>
<td>E-mail address: <a href="mailto:ocorcho@fi.upm.es">ocorcho@fi.upm.es</a></td>
<td>E-mail address: {<a href="mailto:jun.zhao@zoo.ox.ac.uk">jun.zhao@zoo.ox.ac.uk</a>, <a href="mailto:david.deroure@oerc.ox.ac.uk">david.deroure@oerc.ox.ac.uk</a>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Poznań Supercomputing and Networking Center</th>
<th>Instituto de Astrónfisica de Andalucía</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Services Department</td>
<td>Dpto. Astronomía Extragaláctica</td>
</tr>
<tr>
<td>Poznań Supercomputing and Networking Center</td>
<td>Instituto Astrofísica Andalucía</td>
</tr>
<tr>
<td>Z. Noskowskiego 12/14, 61-704 Poznan</td>
<td>Glorieta de la Astronomía s/n 18008 Granada,</td>
</tr>
<tr>
<td>Poland</td>
<td>Spain</td>
</tr>
<tr>
<td>Contact person: Dr. Raúl Palma de León</td>
<td>Contact person: Dr. Lourdes Verdes-Montenegro</td>
</tr>
<tr>
<td>E-mail address: <a href="mailto:rpalma@man.poznan.pl">rpalma@man.poznan.pl</a></td>
<td>E-mail address: <a href="mailto:lourdes@iaa.es">lourdes@iaa.es</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Leiden University Medical Centre</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Department of Human Genetics</td>
<td>Instituto de Astrónfisica de Andalucía</td>
</tr>
<tr>
<td>Leiden University Medical Centre</td>
<td>Dpto. Astronomía Extragaláctica</td>
</tr>
<tr>
<td>Albinusdreef 2, 2333 ZA Leiden</td>
<td>Instituto Astrofísica Andalucía</td>
</tr>
<tr>
<td>The Netherlands</td>
<td>Glorieta de la Astronomía s/n 18008 Granada,</td>
</tr>
<tr>
<td>Contact person: Dr. Marco Roos</td>
<td>Spain</td>
</tr>
<tr>
<td>E-mail address: <a href="mailto:M.Roos1@uva.nl">M.Roos1@uva.nl</a></td>
<td>Contact person: Dr. Lourdes Verdes-Montenegro</td>
</tr>
<tr>
<td></td>
<td>E-mail address: <a href="mailto:lourdes@iaa.es">lourdes@iaa.es</a></td>
</tr>
</tbody>
</table>
## Change Log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Amended by</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>26-06-2013</td>
<td>Rafael González-Cabero</td>
<td>Initial version with the deliverable structure</td>
</tr>
<tr>
<td>0.2</td>
<td>27-06-2013</td>
<td>Rafael González-Cabero</td>
<td>Recommender Service section added</td>
</tr>
<tr>
<td>0.3</td>
<td>23-07-2013</td>
<td>Raul Palma</td>
<td>RO Evolution section added</td>
</tr>
<tr>
<td>0.4</td>
<td>27-07-2013</td>
<td>José Manuel Gómez-Pérez,</td>
<td>Collaboration Spheres section added</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aleix Garrido</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>29-07-2013</td>
<td>Rafael González-Cabero</td>
<td>Version delivered for QA to UNIMAN</td>
</tr>
<tr>
<td>0.6</td>
<td>30-07-2013</td>
<td>Rafael González-Cabero</td>
<td>Updates related with QA</td>
</tr>
<tr>
<td>1.0</td>
<td>31-07-2013</td>
<td>José Manuel Gómez-Pérez</td>
<td>Final corrections</td>
</tr>
</tbody>
</table>
Executive Summary

This deliverable thoroughly describes the current state of the prototypes and models that provide support for the maximization of the sharing and reuse of preserved workflow-centric Research Objects (ROs).

First it contains the description of the work related to evolution model, which enables the representation of the different stages of the RO lifecycle, their dependencies, as well as the corresponding versions of ROs and their aggregated resources. That includes:

- The roevo ontology, which complements the RO model and enables the formal representation of the identified stages of the ROs lifecycle, along with the dependencies that may exist between ROs and their constituent resources.
- An evolution API for the RO model, and software for supporting the RO evolution and visualization software that depict graphically RO versions dependencies.

Second, it presents the Recommender Service prototype, a software component that uses different data sources and the RO model, provides recommendations to Wf4Ever users about ROs, research colleagues and scientific content. Its key characteristics are:

- The use of different recommendation algorithms, so that it can complement its different features, lessening their drawbacks and enabling a novel policy-based customization of its recommendations. This mechanism will allow us to tailor the Recommender Service to different research fields and different communities of researchers.
- The inclusion of an inference engine that uses the RO model for the deduction of new recommendations. The inclusion of the inference engine along with the creation of the roevo ontology brings also the possibility of recommending researchers with the existence of newer versions of the scientific content that they handle.

Third it describes the Collaboration Spheres mechanism, which provides an instrument to improve, sharing and reuse of ROs and users experience. It is based on:

- The use of advanced visualization techniques that integrate search and recommendation techniques, allowing thus the end user to interact in an intuitive way benefiting from the Recommender Service results.
- The exploitation semantic descriptions and similarity relations between ROs and users.

This deliverable supersedes D3.2v1 with regard to the information about WP3 models and components, and should be read in tandem with D1.2v2, D1.4v2, D2.2v2, and D4.2v2, so as to get a complete overview of the current state of the components and models implemented so far in the Wf4Ever project.
Table of contents

Wf4Ever Consortium .................................................................................................................. 2
Change Log .............................................................................................................................. 3
Executive Summary ................................................................................................................... 4
Table of contents .................................................................................................................... 5
List of Figures .......................................................................................................................... 7
Introduction ............................................................................................................................. 8

1 Research Object Evolution Model ....................................................................................... 9
  1.1 Introduction ..................................................................................................................... 9
  1.2 ROs Lifecycle .................................................................................................................. 9
  1.3 Roevo ontology ............................................................................................................ 13
  1.4 Technological support ................................................................................................... 15
    1.4.1 Research Object Evolution API ............................................................................... 15
    1.4.2 Managing RO evolution from RO Portal ............................................................... 15
    1.4.3 Managing RO evolution from RO Manager ......................................................... 17
    1.4.4 Sequence diagrams ............................................................................................... 17

2 Research Object Recommender Service ............................................................................. 20
  2.1 Introduction ................................................................................................................... 20
  2.2 Recommender Service Architecture ............................................................................ 21
  2.3 Recommenders and Inference Engine ......................................................................... 23
    2.3.1 Batch Recommenders and Inference Engine ....................................................... 25
      2.3.1.1 Collaborative Filtering Recommender ......................................................... 25
      2.3.1.2 Social Network recommender ................................................................. 26
      2.3.1.3 Keyword Content-Based Recommender .................................................. 28
      2.3.1.4 Inference Engine ......................................................................................... 29
      2.3.1.5 Batch Recommenders and Inference Engine Interactions ......................... 31
    2.3.2 On The Fly recommenders Description .................................................................... 33
      2.3.2.1 On The Fly Recommenders Interactions ..................................................... 33
  2.4 Recommender Service API ............................................................................................ 35

3 Collaboration Spheres .......................................................................................................... 41
  3.1 Introduction ..................................................................................................................... 41

2013 © Copyright lies with the respective authors and their institutions.
List of Figures

Figure 1 RO lifecycle scenarios ................................................................. 10
Figure 2 A sample RO lifecycle ................................................................. 12
Figure 3 Main classes and properties of roevo ontology .............................. 14
Figure 4 Interface for creating RO Snapshot or RO Archived in RO Portal ...... 16
Figure 5 RO evolution history in RO Portal ................................................ 17
Figure 6 The client makes a Snapshot RO .................................................. 18
Figure 7 RODL creates a copy of the RO ..................................................... 19
Figure 8 RODL freezes RO copy ............................................................... 19
Figure 9 Recommender Service abstract architecture .................................. 21
Figure 10 Recommenders taxonomy .......................................................... 24
Figure 11 Graphs involved in the Interaction similarity ................................ 28
Figure 12 RecommendationsSetResource creation ..................................... 32
Figure 13 ContextRecommendationsSetResource context creation ................ 34
Figure 14 RORecommendationsSetResource creation .................................. 35
Figure 15: Collaboration Spheres - Sequence Diagram and API .................... 45
Figure 16 Collaboration Spheres at Wf4Ever Sandbox with MyExperiment data .................................................. 46
Figure 17 Collaboration Spheres for APA demo ......................................... 48
Introduction

The main objective of WP3 in the Wf4Ever project is to provide adequate means to maximise the share and reuse of preserved workflow-centric ROs [2] [3] (ROs from now on), while supporting their evolution and versioning; enabling thus the collaboration among scientists.

In order to achieve such objective we propose a social approach that takes advantage of social features stemming from social networks, and we complement them with formal models considering RO evolution at the core of the proposed model. The outcomes of our research in WP3 are:

- **RO collaboration and evolution model.** A model that provides a precise description of the evolution, tracing precisely the progress of a RO; and enables the collaboration among researchers in the creation of ROs as it provides means for describing mixed stewardship situations.

- **RO Recommender Service.** There is a high risk that valuable information could end unnoticed whilst researchers are overwhelmed with peripheral information. The Recommender Service brings useful hints to the researcher in a proactive manner providing, without prior request by the user, practical suggestions of ROs, scientific content and data, publications etc., even in cases where the user may not be aware of their existence beforehand.

- **Collaboration Spheres.** Collaboration Spheres provide a mechanism to improve, share and reuse ROs and user experiences based on the exploitation of semantic descriptions, relations and similarities between ROs and users, in order to support advanced search through visualization mechanisms.

We describe the Wf4Ever Phase II prototypes and draft specifications associated with each of these outcomes in the rest of the document.
1. Research Object Evolution Model

1.1 Introduction

RO evolution, in the scope of this work, refers to the ability of managing changes in a RO and its aggregated resources by creating and maintaining different versions of the RO during its lifecycle (see Section 1.2). It provides a detailed description of the changes in these resources, tracking contributions reused from other sources, as well as attribution information of the agents responsible for their existence. It, thus, enables tracing the progress of a RO and accessing concrete versions of a RO (and their aggregated resources) or individual resources (notably workflows).

In our context we consider an RO version as a specific form or variant of a RO that is normally created after applying changes to an existing variant. A version has an associated state, e.g., live, archived, and they are related to each other via direct relationships (e.g., wasRevisionOf) or via contribution dependencies (e.g., derivedFrom, relatesTo).

1.2 ROs Lifecycle

The lifecycle refers to the stages that the RO transition from its conception until its conclusion. After extensive analysis and discussions with our users, we have identified the following stages for a RO:

- **Live ROs**: represent a work in progress. They are thus mutable as the content or state of their resources may change.
- **Snapshot ROs**: are intended as a record of past activity, ready to be disseminated as a whole. They are immutable, and reflect the state of the Live RO at a certain time.
- **Archived ROs**: represent the final stage of a RO where it has either reached a version that the author prescribes to be stable and meaningful and is appropriate for publication and long-term preservation, or it has been deprecated. They are therefore immutable, with no further changes or versions allowed.

Figure 1 illustrates different lifecycle scenarios for ROs. The life of a RO starts with the creation of a Live RO. While working with the Live RO, i.e., during the maturation of a Live RO while a researcher is designing and executing his investigation, zero to many Snapshot ROs are produced for different purposes (non exhaustive list). When a RO reaches the end of its life, an archived RO may be produced. It is the endpoint in the life of a RO. After producing an archived RO, the RO does not evolve further; however earlier stages of the RO may have produced other lines of work (forks) creating new Live ROs. Table 1 summarizes the characteristics of ROs in each of the three stages.
Figure 1 RO lifecycle scenarios

Typically, an archived RO is a stable and meaningful version appropriate for public release and long-term preservation. However, alternatively, an archived RO could be created when the RO has been deprecated and its development has been stopped. In a typical scenario, a Live RO gives raise to a number of Snapshot ROs before producing an Archived RO. However, the Archived RO may be reused later by the original producers or by third parties in order to extend the associated investigation or to start a new investigation based on the results produced by the RO, thereby creating a new Live RO.

Hence, as depicted in Figure 1, a new RO may be created:

- From scratch.
- From a set of existing resources.
- By forking a Live RO, e.g., split/replicate a Live RO in order to explore different hypothesis.
- By forking from a Snapshot RO, e.g., recover and re-use a frozen copy of the RO at a certain time, to explore different hypothesis.
- By reusing an Archived RO, e.g., continuing the research after an Archived RO is produced to produce new results and make progress in the research.

The new ROs should keep the history of its ancestors when it is created from forking or re-using other resources. This history is particularly interesting to the users, e.g., to visualize the evolution of the RO integrity, stability, completeness, quality, etc. Note that ROs are potentially under the control of multiple owners falling under mixed stewardship, and they (or its aggregated resources) may be shared among a
restricted group of users, raising issues of security and access control; these issues are addressed differently depending on the collaborative and versioning tools used for maintaining them (e.g., digital library, SVN, Dropbox, etc.).

Table 1 Characteristics of ROs in Different Stages

<table>
<thead>
<tr>
<th>RO Type</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Live ROs</td>
<td>• They are private to the researcher or they are shared with other collaborators via mail, in a common space/server, or with the help of collaborative tools like Dropbox, SVN, Git, digital libraries, etc. Thus, a Live RO could be also a set of resources, with a structure that may not adhere to the RO model.</td>
</tr>
<tr>
<td></td>
<td>• They may be versioned with tools like Git or SVN (e.g., as it happens with educational resources as described in <a href="http://mashe.hawksey.info/2012/03/do-you-git-it-open-educational-resource-practices-meets-software-version-control/">http://mashe.hawksey.info/2012/03/do-you-git-it-open-educational-resource-practices-meets-software-version-control/</a>)</td>
</tr>
<tr>
<td></td>
<td>• A Live RO reaches a certain point in time during its evolution when it needs to be disseminated or preserved and consequently a Snapshot RO or Archived RO is produced.</td>
</tr>
<tr>
<td>Snapshot ROs</td>
<td>• They are shared among a group of selected users, e.g., with the research group for internal review or with external referees for external review; they are not publicly available by default.</td>
</tr>
<tr>
<td></td>
<td>• They are related to each other via versioning relations.</td>
</tr>
<tr>
<td></td>
<td>• They are usually stored in a digital library (e.g., RO Digital Library - RODL¹), which is the responsible for their versioning.</td>
</tr>
<tr>
<td></td>
<td>• They must be referenceable.</td>
</tr>
<tr>
<td></td>
<td>• They serve as a frozen copy of the state of an investigation that may be preserved for the researcher and/or research team.</td>
</tr>
<tr>
<td></td>
<td>• They serve for internal or external review (e.g., between supervisor and student, or between researcher and referee). The evaluation of the snapshot is then used for the further development of the Live RO. Typically, the snapshot RO is kept by the researcher for later reference.</td>
</tr>
<tr>
<td></td>
<td>• They serve as the basis for reaching an archived RO.</td>
</tr>
<tr>
<td>Archived ROs</td>
<td>• They are typically publicly available, but they may be restricted to a certain group of people.</td>
</tr>
<tr>
<td></td>
<td>• They are related to the previous snapshot created from the same RO (if available) via versioning relations.</td>
</tr>
<tr>
<td></td>
<td>• They are usually stored in a digital library (e.g., RODL or Journal DL infrastructure).</td>
</tr>
<tr>
<td></td>
<td>• They should be curated (see <a href="http://www.dcc.ac.uk/digital-curation/what-digital-curation">http://www.dcc.ac.uk/digital-curation/what-digital-curation</a>).</td>
</tr>
<tr>
<td></td>
<td>• They must be citable.</td>
</tr>
<tr>
<td></td>
<td>• They must be preserved.</td>
</tr>
</tbody>
</table>

¹ [http://www.wf4ever-project.org/rodL](http://www.wf4ever-project.org/rodL)
Illustrative example

We will now illustrate RO lifecycle through a small example that shows how all the resources contained in a RO are bundled as the scientific experiment progresses. This example lifecycle is summarized graphically in Figure 2.

Figure 2 A sample RO lifecycle

An RO starts its life as an empty Live RO, with a first design of the experiments to be performed. Gradually, it is incrementally filled within the underlying system by aggregating datasets, documents, workflows encoding scientific methods, and other related resources. These resources may be generated from scratch or by reusing and repurposing existing resources, keeping the record of contributions and attributions. This information may be generated automatically by the underlying system (e.g., the original source), or provided by the user (e.g., derived from). While working with the Live RO, its resources may be changed at any point in time, new resources may be added, other may be removed, and they may be annotated. Moreover, executable resources such as workflows may be run several times using different inputs and producing different outputs, generating provenance information of the results that may be recorded in the underlying system.
In our example, we observe several points in time when this Live RO gets copied to produce a RO Snapshot, which aims to reflect the status of the RO at a given point in time. Such a snapshot may be useful to release the current version of the research outcome of an experiment, submit it to be peer reviewed or to be published (with the appropriate access control mechanisms), share it with supervisors or collaborators, or for acknowledgement and citation purposes. A snapshot may also contain a paper describing the RO in general, and the experiment in particular, depending on the policies of the corresponding scientific communication channel, e.g., workshop, conference or journal.

Each Snapshot has versioning information associated (e.g., previous version) usually generated by the underlying system, including when this Snapshot was created, from which Live object is derived, by whom and how it was changed from the previous version (e.g., the set of changes). Changes and versioning information may be associated to the whole RO and its aggregated resources. Snapshots have their own identifiers, and may be preserved, since it may be useful to be able to track the evolution of the RO over time, so as to allow, for example, retrieval of a previous state of the RO, reporting to funding agencies the evolution of the research conducted, etc. Additionally, Snapshots may have associated quality-related information, providing useful information of how the RO quality has changed throughout its lifetime.

At some point in time, the RO may get published and archived, in what we know as an Archived RO. It has a permanent identifier and may be created by copying completely the Live RO, or it may be the result of some filtering or curation process where only some parts of the information available in the aggregation are actually published for others to reuse. Later, as illustrated in Figure 2 an Archived RO can be used as a starting point for a new research work, e.g., by repurpose the whole RO or its parts, in which case it is used as the base to create a new Live RO.

1.3 roevo ontology

The RO evolution model enables the representation of the different stages of the ROs lifecycle, their dependencies, as well as the corresponding versions of ROs and their aggregated resources, with the associated changes in these resources. The concrete realization of this model is the roevo Ontology\(^2\), which is built on top of the core ro ontologies (see D2.2v2).

The latest version (v_0.4) had two main goals for its design: (i) to align it with the provenance vocabulary (PROV ontology - [https://dvcs.w3.org/hg/prov/raw-file/default/ontology/Overview.html](https://dvcs.w3.org/hg/prov/raw-file/default/ontology/Overview.html)) and (ii) to simplify the taxonomy of changes based on user feedback. On the one hand, we reused and extended when appropriate classes and properties from the PROV ontology. On the other hand, we simplified changes into additions, removals and modifications, which are the ones absolutely required by users, and left the option to create extensions of roevo to model more detailed taxonomies of changes for required resources, such as workflows and annotations.

The main classes and properties of roevo are depicted in Figure 3. The ontology extends the ro ontology by defining subclasses for the different states of a RO. In particular, it defines LiveRO, SnapshotRO and

\(^2\) Available at: [http://purl.org/wf4ever/roevo](http://purl.org/wf4ever/roevo)

2013 © Copyright lies with the respective authors and their institutions.
ArchivedRO subclasses. Moreover, these classes are also defined as subclasses of the Entity class of PROV ontology, as these are the objects we want to provide provenance for. In order to specify the agent responsible for the creation of a SnapshotRO or archivedRO, they are related to the Agent class of the PROV ontology via the relations wasSnapshotedBy and wasArchivedBy, respectively. In order to keep the versioning of ROs and their aggregated resources, roevo also defines the class VersionableResource as the union of SnapshotRO, ArchivedRO and the class Resource from ro ontology. This class is also defined as subclass of Entity class of PROV ontology, and it allows keeping the history of versions via the property wasRevisionOf. Similarly, in order to specify the set of changes between versions, we defined the class ChangeSpecification and associated it with the Change class to specify the individual changes. Both of these classes are defined as subclasses of the Activity class from PROV ontology, which refers to something that occurs over a period of time and that acts upon or with entities (ROs or their aggregated resources in our case). The class Change has three subclasses for the general type of changes required by our users (i.e., Addition, Modification and Removal). More detailed changes can be specified by providing specializations for these classes, in particular of the Modification class. For example, an extension for workflow changes defines particular workflow changes, such as AddProcessor, AddWorkflowPort, AddControlFlowBetweenProcessors, etc. Finally, to express contribution relations, we rely on the property of PROV ontology wasDerivedFrom and its subproperties.

As we have already stated, roevo ontology is available at http://purl.org/wf4ever/roevo. Additional information and links to the extensions are available at http://www.wf4ever-project.org/wiki/display/docs/RO+evolution.

![Figure 3 Main classes and properties of roevo ontology](image-url)
1.4 Technological support

The roevo ontology has been used as the underlying model for the definition of the evolution API. RO Digital Library – RODL implements this API and two of its clients, namely RO Portal and RO Manager, expose the API functionalities to the users, enabling them to manage evolution of ROs from a web user interface and a command line interface, respectively.

1.4.1 Research Object Evolution API

The goals of this API are to enable transformation of the ROs based on their lifecycle and to facilitate retrieving the evolution history of the ROs. It is based on the roevo ontology.

The lifecycle transformation of ROs is achieved by providing a service that performs a copy of a RO and saves the copy as a live, snapshot or archived RO. The client needs to make two requests to complete the copying:

- **Copy a RO.** The new RO is in a transient state and is available only to the creator. It can still be modified before reaching the target evolution state.

- **Finalize the state transformation.** The service validates that the transient RO meets the requirements for being in the target evolution state and performs the state transition. For evolution states such as a snapshot or archived RO, that means that the RO becomes immutable.

The first and second requests can be combined into one if the client does not intend to modify the copy before finalizing the transformation.

The second API functionality is a facility for retrieving the evolution provenance. The client may send a request with a RO URI as a query parameter and the service will return an RDF graph with selected information about the RO in question, its copies and the RO that it itself was copied from. Note that typically this information can equally be achieved with a series of SPARQL queries to the ROSR service hosting the RO.

A detailed description of the operations and additional considerations of the API can be found in D1.3v2 and is also available online at [http://www.wf4ever-project.org/wiki/display/docs/RO+evolution+API](http://www.wf4ever-project.org/wiki/display/docs/RO+evolution+API).

1.4.2 Managing RO evolution from RO Portal

Currently, users are able to create RO snapshots or RO archives in RO Portal from a Live RO stored in RODL. For this task, users open a Live RO in RO Portal and selects snapshot or release (archive) button from the Evolution dropdown list in the Overview tab, as depicted in Figure 4, and the RO Snapshot or RO Archive is created in the background. Note that the current implementation in RO Portal performs two operations in one step when selecting the snapshot/archive button, i.e., the creation of the RO copy and the finalization of the state transformation.

---

3 [http://www.wf4ever-project.org/wiki/display/docs/Research+Objects+Digital+Library+%28including+the+ROSRS%29](http://www.wf4ever-project.org/wiki/display/docs/Research+Objects+Digital+Library+%28including+the+ROSRS%29)

2013 © Copyright lies with the respective authors and their institutions.
Additionally, users can inspect the evolution of ROs in RO Portal from the RO History tab, as depicted in Figure 5. This tab shows a graph of ROs or similar resources that are related to the inspected RO in terms of evolution. If the current RO is a snapshot, the graph may show its live RO and other snapshots, for example. The graph is built using the information returned by RODL, which is based on roevo ontology. The inspected RO is always marked with a blue background. Although at the moment Snapshot and Archived ROs can be created only from Live ROs stored in RODL, in the graph not all resources must be ROs in a strict sense, for example the Live RO may be a URI pointing to a stack of files with no metadata. In the future, RO Portal will enable the creation of snapshots and archives from Lives ROs stored somewhere else. A complete user guide of RO Portal can be found in D2.1v3.

Figure 4 Interface for creating RO Snapshot or RO Archived in RO Portal
1.4.3 Managing RO evolution from RO Manager

RO Manager is a command line tool that enables access and use of ROs and aggregated resources stored on the local file system, and their synchronization with remote repositories, e.g., RODL. RO Manager allows creating an immutable snapshot or archive of a RO in a repository implementing the RO Evolution API (e.g., RODL). The local RO must be first pushed (i.e., synchronized) to such repository before it can be snapshotted or archived. For these evolution tasks, RO Manager provides three commands:

- “snapshot”, which prepare a snapshot of an existing RO
- “archive”, which prepare an archive (also called release) of an existing RO. This command is similar to “snapshot” but it creates an immutable RO with a status of “archive” rather than “snapshot”.
- “freeze”, which freezes a previously prepared snapshot or archive making it immutable.

Note that between the “snapshot” and “freeze” operation the RO can be still modified, for example, to curate it or selecting the resources that want to be published. A complete user guide of RO Manager can be found in D2.1v3 and D2.1v2.

1.4.4 Sequence diagrams

Figure 6 presents the sequence of actions required to make a snapshot or archive of a RO, as seen by the client. Creating a snapshot is asynchronous, which means that the client first creates a job and then it can monitor it until it is finished. The first job makes a copy of a RO, which can later be modified. When the copy is ready to become an immutable snapshot or archive, the client issues a freeze request. This creates a second job. When it finishes, the snapshot or archive is ready.
Figure 6 The client makes a Snapshot RO

Figure 7 presents the sequence of actions performed by RODL in order to make a mutable copy of a RO (first job in first paragraph), which will become a snapshot or an archive. This is an asynchronous operation, which means the request from the client is not synchronized with it. After the request dispatcher starts the
copy operation, a new RO is created and each resource is copied. Finally, the evolution information is generated for the copy, which specifies the relations between the copy and the original.

Figure 7 RODL creates a copy of the RO

Figure 8 presents the actions performed by RODL when the client creates a job of freezing a RO copy, the so-called “freeze” or “finalize” operation (second job in first paragraph). The request dispatcher forwards the request to the RO model instance, which sets appropriate evolution status and stores it.

Figure 8 RODL freezes RO copy
2. Research Object Recommender Service

1.5 Introduction

The Recommender Service brings useful hints to the researcher in a proactive manner providing, without prior request by the user, practical suggestions of scientific data and results in the shape of:

- **ROs recommendations.** The whole bundle of experimental content is handled to the user as probable relevant material.

- **Scientific resources recommendations.** When making recommendations to a given user the system may also suggest isolated resources that might be useful additions/alternatives to the ones already aggregated by the ROs that the user is currently using or creating.

The advantage of the recommendation activity over search mechanisms in the scientific content discovery scenario that recommender systems do not presume that the user has the notion relevant scientific content existence (i.e. the user can be unaware of relevant information). The Recommender Service performs such recommendation activity using novel multi-faceted recommendation techniques; mixing community-based and advanced demographical/content-based recommendations along with collaborative filtering techniques. More precisely:

- **Different facets** of users and ROs (they will be described later on when describing the recommenders).

- **Different relevance measures and policies** regarding the significance of scientific content in different fields or research communities.

In this document we describe the Wf4Ever Phase II prototype of the Recommender Service:

- It is **fully functional** and has been deployed and it is up and running as part of the Wf4Ever Toolkit in the Wf4Ever sandbox\(^4\) (see [8] for a complete and up to date description of the Wf4Ever Toolkit).

- It has been **integrated in** the alpha version of myExperiment\(^5\), where it is publicly available for users outside of Wf4Ever researchers.

- It has been **integrated** with the **Collaboration Spheres** in order to provide a graphical approach to the examination of recommendations results.

- Its **source code** is publicly available at GitHub\(^6\), available under the terms of the MIT open source license\(^7\).

---

\(^4\) http://sandbox.wf4ever-project.org/recommenderService
\(^5\) http://www.myexperiment.org
\(^6\) https://github.com/wf4ever/epnoi
\(^7\) http://opensource.org/licenses/MIT
• The **details** of its interface, data format, configuration and deployment, etc. are available at the Wf4Ever public wiki pages⁸

### 1.6 Recommender Service Architecture

The architecture of the Wf4Ever Phase II prototype of the Recommender Service is depicted in Figure 9.

![Figure 9 Recommender Service abstract architecture.](image)

Let us describe these components, some of which will be described in much more detail in forthcoming sections. They are:

---

⁸ [http://www.wf4ever-project.org/wiki/display/docs/Recommender+Service](http://www.wf4ever-project.org/wiki/display/docs/Recommender+Service)

2013 © Copyright lies with the respective authors and their institutions.
• **Core.** The Recommender Service core is the main module of the Recommender Service architecture. Its main functions are:
  
o To provide an interface to the recommender external components (the Recommender Service API and Recommender Service homepage).
  
o To initialize and orchestrate the rest of the elements that provide the Recommender Service functionality. That includes the initialization and handling of all the recommenders (and its different parameterizations) when the Recommender Service boots.

• **Shared Model.** The Shared Model acts as a facade for accessing to all the considered content for supplying recommendations, which at the moment includes information about all myExperiment users, files, and workflows; and the information about scientific papers that is extracted from different OAI-PMH\(^9\) repositories. Underneath this virtualization layer different storage and indexing systems are situated; each of which optimized for different purposes. Briefly they are:
  
o **Apache SOLR.** SOLR is an open source enterprise search platform under the umbrella of the Apache Lucene project\(^{10}\). It provides in a scalable and fault tolerant way full-text search, faceted search, highly efficient indexing, dynamic clustering, etc. The Recommender Service stores the text related with user's profiles, workflows, files, and approximately a million of research papers abstracts and related information in an Apache SOLR instance.
  
o **Neo4J.** Neo4j is an open source transactional graph-oriented database\(^{11}\). In the Recommender Service is used to store all the social information about users (i.e. users, user's groups and the relationships among them) as it is heavily optimized to process queries that traverse graph-structured data.
  
o **Apache Cassandra.** Apache Cassandra is an open source distributed database management system. It is a NoSQL database described as a BigTable data model running on an Amazon Dynamo-like infrastructure\(^{12}\). Cassandra provides a structured key-value store with a data consistency level that can be tailored to the needs of each application. The Recommender Service uses it to store in memory all the necessary data to provide recommendations, which includes information about users (except for the social aspects that we afore described), recommendations and its provenance, information about external resources, and a long etc.

• **Harvesters and indexers.** In order to obtain recommendations, the Recommender Service uses different harvesters and indexers that extract and introduce relevant information in the Shared Model so that the recommenders can use it. Currently we support access to myExperiment, OAI-PMH compliant scientific repositories and e-print services, and RSS feeds.

---

\(^9\) [http://www.openarchives.org/pmh/](http://www.openarchives.org/pmh/)

\(^{10}\) [http://lucene.apache.org/solr/](http://lucene.apache.org/solr/)

\(^{11}\) [http://www.neo4j.org/](http://www.neo4j.org/)

• **Recommenders.** As we described in the introduction of the Recommender Service description, our aim is to provide a recommender system that takes into consideration different users and ROs facets; and different relevance measures and policies. To do so, the Recommender Service hosts a family of different recommenders that use different recommendation algorithms with different parameterization. The description of these components and the algorithms that they use will take most of the space of the rest of the Recommender Service description.

• **Recommendations Spaces.** Recommendations Spaces aggregate recommendations and provide multifaceted retrieval functions for them. There are two predefined Recommendations Spaces, one is the main Recommendation Space, which is used to store the recommendations created by the family of recommenders in the system; and the other, the inferred Recommendation Space, which is created to store the recommendations inferred from the former (we provide more details about this in section 1.7.2.1 On The Fly Recommenders Interactions). As in the case of the Shared Model, the Recommendations Spaces is in reality a facade to provide simplified access recommendations and associated data. The actual data is efficiently stored in memory in an Apache Cassandra instance.

• **Inference Engine.** The Inference Engine infers new recommendations from previous ones created by the set of recommenders present in a Recommender Service instance. It introduces knowledge based techniques and ontologies, which as we shall describe, brings many benefits lessening the effects of the dependence on statistical data of many recommendation techniques.

• **Recommendations combiner.** Recommender systems are inherently vertical and configured to provide recommendations in a single and specific domain. We need of means for tailoring specific recommendations in terms of each research community that in the future wishes to make use of the recommender system. We address this tailoring activity when we combine the recommendations obtained with different recommendation algorithms that we have described. The current prototype of the Recommender Service provides a straight-weighted results combination that can be configured by the user; we leave the use of the complex combination policies for future work. Regarding the user requirements described in [9] the recommendations combiner module on the one hand aims at lessening the afore-described problems associated with each type of recommender, and on the other hand tackles the policy-based recommendation requirement.

• **REST API.** The Recommender Service exposes its functionality as a fully REST compliant interface. We provide a detailed description of such interface in the section 1.8 Recommender Service API.

### 1.7 Recommenders and Inference Engine

As we have already stated, and as is shown in Figure 9, the Recommender Service uses a variety of recommenders, each of which with its own recommendation algorithms and parameterization (i.e. we consider the case of more than one recommenders using the same recommendation algorithms with different parameters).
In Figure 10 we illustrate the taxonomy of recommenders present in the current implementation of the Recommender Service.

We have classified and implemented them according to their interaction nature, the algorithm they use, and the type of the items that they recommend. Thus we have:

- **According to their nature.** Recommenders can be either BatchRecommenders, recommenders that need significant time to carry out their algorithms and henceforth their recommendations need to be calculated before requests for recommendations are handled; and OnTheFlyRecommenders, which are those that perform their recommendation algorithm and calculations just in time when the requests arrive.

- **According to their algorithm.** We distinguish between collaborative filtering recommenders, content-based recommenders, RO-based recommenders, context-based recommenders, and finally social recommender.
• According to the recommended item. That includes myExperiment users, files and workflows; and external research resources extracted from OAI-PMH repositories and e-print services.

1.7.1 Batch Recommenders and Inference Engine

1.7.1.1 Collaborative Filtering Recommender

Collaborative filtering techniques (e.g. [10] [14] [15] [11]) predict user’s affinity for items on the basis of the ratings that other users have made to these items in the past. Therefore, the steps taken to make recommendation in such systems consist in finding people with similar tastes to the user (or items with similar rating patterns as the one that the user has rated) by means of its past ratings; and by means of their ratings extrapolate the user future ratings. User information in a collaborative system consists of a vector of items and their associated ratings; finding similar users translates into finding similar vectors. The main advantages of collaborative techniques are:

• **Cross-genre niches identification.** Collaborative filtering has proven to be very effective at thinking out-of-the-box, relating what apparently is not related.

• **Domain independence.** Domain knowledge is not needed (e.g. the same algorithm that rates movies can be used to recommend ROs).

• **Improves over time.** The quality of its results improves over time just taking into and implicit user feedback sufficient.

Unluckily, the use of collaborative filtering techniques is also accompanied of some drawbacks. The main disadvantages of collaborative filtering are due their dependence on large historical data set for getting good quality results, causing:

• **Cold-start problems.** They are:

  o **New user problem.** When a new user arrives at the system, there is no sufficient rating information to sketch user’s preferences; and there might be also a lack of information about the user itself.

  o **New item problem.** Every time a new RO is created the Recommender Service must recommend this new item and make to any of the users of the system that might be interested on it. Unlike the case of the new user problem, the possibility of not having enough information about the RO is less probable, since we assume that the information about the RO is openly accessible, but yet possible.

• **Gray sheep problem.** This problem is related with the new user problem. Some users are mere observers in a social scenario; they don’t rate items nor provide any means to extract their taste form their social interactions. Therefore, the system does not posses enough information about them, and therefore their interests are hard to characterize.
• **The sparsity problem.** The sparsity problem typically occurs in systems with large number of items in which there are plenty of items rated only by few users, and many users, which rated only few. The set of items rated but just few users would unlikely be recommended, no matter how high its reputation might be.

The algorithm that the Collaborative Filtering recommender of the Recommender Service performs for each user $u$ consists in the following steps:

- **Step A.** *User u neighbourhood determination.* We refer as the neighbourhood of a user to the set of users that are similar her. As we are describing a collaborative filtering algorithm, user similarity means similarity in tastes, which is determined by the past ratings of users to the items in the system (we use the past ratings of the user in the myExperiment portal). We allow two types of neighbourhoods, which can be chosen when configuring the instance of the recommender:
  
  - One defined by a constant $Max$ that represents the fixed size of the neighbourhood (we choose up to $Max$ similar users in the neighbourhood, ordered by their similarity with the user)
  
  - One defined by a constant $NeighbourhoodThreshold$ that represents the similarity threshold (we consider as part of the neighbourhood to those users at least $NeighbourhoodThreshold$ similar to user $u$)

- **Step B.** *Inference of new ratings values and creation of recommendations.* For each item $i$ that:
  
  - User $u$ has not yet rated.
  
  - At least one of the $n_j$ in the neighbourhood has rated $i$ in the past with an strength $rating(n_j,i)$. We note the set of all the neighbours that have rated this item as $NeighbourhoodRated(u,i)$

We infer the rating that $u$ would give to item $i$. We note it as $s$, and is defined as a similarity-weighted average of its neighbourhood rating values (Note that we normalize ratings value to the $[0,1]$ range)

$$s = \frac{1}{|NeighbourhoodRated(u,i)|} \sum_{n_j \in NeighbourhoodRated(u,i)} rating(n_j) * similarity(u,n_j)$$

In case that $s$ > threshold (threshold is a constant parameter of the algorithm) we create a recommendation for the user $u$, of the item $i$, with a strength $s$.

With the inclusion of the Collaborative Filtering recommender we address the discoverable RO requirement and, more importantly, the reputation requirement from the catalogue of requirements that were identified in [9] for the Recommender Service.

1.7.1.2 **Social Network recommender**

Social recommenders provide recommendations of users on the basis of their interactions with other users; both from a social perspective (i.e. interactions with other users that have been previously labelled as friends by the user); and authorship network perspective (i.e. other user’s that have co-authored scientific content in
the past). In this first version of the recommender we only consider the first type of interactions, those rooted in its social network.

- **Step A.** *The creation of the social network graph.* This graph is defined using the friendship and group belonging relationships that are defined in myExperiment\(^\text{13}\). The nodes represent users; the edges of the graph represent friendship and same group belonging.

- **Step B.** *Use similarity measurement.* Once we define the graph, we use the similarity measure between an user \(u\) and user \(x\) defined in [1]:

\[
\text{InteractionSimilarity}(u,x) = \log \left( \frac{\text{MFG}(u,x).E}{\log(2 \cdot \text{FG}(u).E)} \right)
\]

Where:

- **\(\text{FG}(u)\)** is defined as the friends graph of the user \(u\), that is, the part of the whole social graph that contains the users directly connected with the user as nodes, and their relationship as edges. \(\text{FG}(u).E\) represents precisely the set of edges of such subgraph.

\[
\text{FG}(u).E = \{n,n' \mid n,n' \in \text{FG}(u).N, e=(u,n) \}
\]

- **\(\text{MFG}(u,x)\)** is defined as the mutual friends graph that user \(u\) and user \(x\) share. It represents the part of the social graph that contains those users that are directly connected to both user \(u\) and user \(x\).

- **\(\text{MFG}(u,x).E\)** is the shorthand for the set of edges of such graph. Formally:

\[
\text{MFG}(u,x).E = \{n,n' \mid n,n' \in \text{MFG}(u,x).N, e=(u,n) \}
\]

- **\(\text{MFG}(u,x).N\)** is the set of nodes of the mutual friends graph defined as follows (we note as \(G\) to the graph that represents the social network). We represent examples of these subgraphs for the user \(u\) with regard to users \(x\) and \(y\) in Figure 11.

\(^{13}\) http://www.myexperiment.org/
Informally speaking, this measure of similarity compares the number of common friends of the users being compared with those friends that each of them posses. The more similar these sets are, the more similar these users are.

- **Step C. Users recommendations creation.** In case that the $\text{InteractionSimilarity}(u, x) > \text{threshold}$ ($\text{threshold}$ is a constant parameter of the algorithm) we create a recommendation for the user $u$, of the user $x$, with a strength $s$.

With the inclusion of the Social Network recommender we address the social aspects of users that where not identified in [9] as a requirement for the Recommender Service, but that in the course of the project have turn out to be of great importance.

### 1.7.1.3 Keyword Content-Based Recommender

Content-based recommender systems (e.g. [2] [12]) make use of information retrieval and filtering techniques. A content-based recommender tries to infer users future items of interest on the basis of the features of the objects that the users uploaded, handled or rated in the past. These object features are items of interest such as keywords that define the object, a summary of its content, etc.. Content-based techniques have similar advantages to collaborative filtering approaches (without the ability of detecting cross-genre niches), and they do not exhibit the new item problem. Nonetheless, they still rely in a large historical data set.

The advantages of content-based recommendation algorithms are:

- **No new item problem.** Once an item is incorporated to the system, a content-based recommender is able to compare its description with each user profile and measure its relevance for each user.

- **Isolated user benefits.** The information provided by an active user to build her own profile translates into an instant improve of the recommendations made by the system, as there is no need for data from other users.

The main disadvantage of content-based recommendation algorithms is the new user-handling problem. When the user approaches the system for the first time it stills does not have a well-formed user’s profile. Nevertheless, this problem is less acute that in the case of the collaborative filtering techniques, as this technique does not rely on long-time statistical information; it just needs that the user provides a small set of keywords that represent its interests. Content-based recommenders recommend items based upon:

- **A description of the content of the item** (i.e. ROs or resources) The RO (or resource) description, being particularly significant the title, description, and the tags that have been applied to the item by the user community.
- **A profile of the user’s interest.** The user’s profile is embodied by a set of keywords that have been previously proposed by the user (and its assigned tags).

The algorithm that the Keyword Content-based recommender of the Recommender Service performs can be described in the following steps:

- **Step A. Creation of the description of the content of items.** This is mainly a process done in the background when different items (i.e. ROs, RO resources such as papers, workflows, presentations, etc.) are added to the system.

- **Step B. Creation of the user profile of interest.** It is a set of keywords composed by those that the user has previously proposed as a description of her interests and the tags that she has applied scientific content.

- **Step C. Matching process.** For each of the users we match its profile of interest description against each item description. For the matching we use one of the best-known Information Retrieval measures, the **TF-IDF** (*Term Frequency Inverse Document Frequency*) \[16\] a statistical measure valid to evaluate how important a word is to a document in the context of a concrete corpus. In case that for an user \(u\) and for an item \(i\) we the \(TF-IDF(u,i) > \text{threshold}\), (\text{threshold} is a constant parameter of the algorithm) we create a recommendation for the user \(u\) of the item \(i\), with a strength \(TF-IDF(u,i)\).

Finally, from the catalogue of requirements defined in [9] the inclusion of the keyword content-based recommendation algorithm addresses in a generic way the discoverable RO requirement; and more specifically the content-based requirement and partially the cold start requirements.

### 1.7.1.4 Inference Engine

The Inference Engine uses knowledge-based techniques and ontologies to obtain new recommendations from previous ones. We propose the use of the constrained spreading activation mechanism. Constrained activation techniques have been well studied in the Information Retrieval field. Initially defined by [13] [6] . Upon activation of a number of specific nodes, their activation is spread iteratively to adjacent nodes until some termination criterion is met. In the concrete case of the recommendation inference engine the activation equals to item recommendation with a given strength. This knowledge-based technique also brings the benefits of knowledge-based recommenders benefits:

- **Cold start problems lessening.** The Inference Engine relies on the model and reasoning algorithm that it performs. It does not use any kind of statistical data and henceforth can carry out its activities on new items without any problem.

- **External features.** Can include features that are not present in the items (e.g. RO, resource, etc.)

The inclusion of the Inference Engine (or more precisely, the use of the constrained activation technique) is associated with the following drawbacks:
• **Static behaviour.** The propagation of new recommendations is constrained by the relations defined among concepts in the used ontologies. This kind of knowledge is not easily changed (and should not be usually changed).

• **Knowledge engineering required.** The use of the constrained spreading activation mechanism assumes the pre-existence of a formally and explicitly defined model of the domain. In our case this model is the RO model specification that we have defined in the context of Wf4Ever.

Following the approach presented in [7] we adopt a constraint-based approach that introduces path constrains, assigning a weight for each of the relationships that appear in the model; and fan-out constraints, that restrict the propagation of recommendations for those nodes that have a high cardinality neighbourhood.

With the inclusion of the constrained spreading activation technique we effortlessly introduce:

• **Resource aggregation handling.** Once a resource of a RO is recommended, the strength of such recommendation is propagated to the ROs that aggregate such resource. The ROs might be recommended if the recommendation has a high strength or if it contains several resources that are relevant to the user.

• **RO evolution handling.** Using the roevo model the Inference Engine is able to propagate the recommendations made to a concrete version of an RO to other versions of the very same RO that are obviously related.

The Recommender Service Inference Engine that has been implemented and used to recommend aggregated resources. The Recommender Service is able to make recommendations of ROs (currently in the form myExperiment packs) taking into account the recommendations given to its constituent resources (i.e. workflows and files).

The Inference Engine performs a spreading activation algorithm based in the one described in [17] . Let us describe it briefly.

• **Step A. Creation of the activation graph.** We create the activation graph, a directed graph which nodes are classes of the RO model, its edges are the relationships defined in the RO model

• **Step B. Spreading activation.** For each user \( u \) we perform the following activities:
  
  o **Initialization.** We activate the set of nodes that represent the instances of the items that were previously recommended to the user by the set of recommenders in the Recommendation Service. Its activation value is the strength of the recommendation (the rest of the nodes start with an activation value set to zero). We note the activation of each node \( i \) as \( a_i \). We define the total activation constant as the sum of the activations of all the nodes.

  \[
  \text{TotalActivation} = \sum_i a_i
  \]

  o **Activation update.** Each node of the activated nodes updates its activation value. We note \( o_j \) to the output activation of node \( j \) of those connected to \( i \)
\[ a_i = \sum_j o_j \ast w_{ji} \]

where \( o_j \) is the activation of the node \( j \) multiplied by the fan-out factor (one divided by the degree of the node \( j \))

\[ o_j = \frac{1}{\text{degree}(j)} a_j \]

and \( w_{ij} \) represents the weight associated with the

- **Normalization.** We normalize the updated activation of all the nodes taking into account the activation conservation principle. So each activation node is normalized (we note it \( a'_i \)) so that all the activations still sum the \( \text{TotalActivation} \) constant:

\[ \text{TotalActivation} = \sum_i a'_i \]

- **Termination checking.** In case that the activations in the activation graph are stabilized (i.e. there are no significant difference between the new and updated activation values of each node) we stop. Otherwise we keep on spreading the activations.

**Step C. Recommendations creation.** For each user \( u \) we analyze its stabilized activation graph. In case that we find a node \( i \) that signifies an item \( i \) where \( a_i > \text{threshold} \), (\( \text{threshold} \) is a constant parameter of the algorithm) we create a recommendation for the user \( u \) of the item \( i \), with the strength of \( a_i \) normalized in the [0,1] range.

The recommendations inference engine addresses, from the set of requirements defined in [9], the RO evolution aware requirement, the repurposeable ROs requirement, the RO model aware requirement, and partially addresses the Cold-Start requirements.

### 1.7.1.5 Batch Recommenders and Inference Engine Interactions

After introducing the batch recommenders, let us show a high-level bird view of how these main components of the Recommender Service are orchestrated by the Core to provide recommendations. Recommendations, as we shall see in the description of the Recommender Service API in the section 1.8 Recommender Service API, are packed and delivered to client application in the form of bundles that we call resources, RecommendationsSetResource in the case of BatchRecommenders.

For BatchRecommenders the recommendations are provided when the Core is initialized. As described in the sequence diagram the Core starts by creating each of the recommenders (using the RecommenderFactory.buildRecommender method).
After each of the BatchRecommenders have been created and initialized, each of recommender is requested to perform its corresponding recommendation algorithm with its own initialization parameters (these parameters are defined declaratively in an XML configuration file which content is loaded in the ParametersModel object passed as a parameter to the Core initialization method). These recommendations are all stored in the main RecommendationsSpace that the Core holds. After all the BatchRecommenders have provided their recommendations the Core initializes the InferenceEngine. The engine takes the main RecommendationsSpace and creates a new one (which we
refer as the inferredRecommendationsSpace). Recommendations are added to such RecommendationsSpace in the same manner as the recommenders did it. Finally, when the client requests a RecommendationsSetResource, both the recommendations contained in the main recommendationsSpace and in the inferredRecommendationsSpace are retrieved (using the getRecommendationsForUser method that the EpnoiCore exposes). Finally, with these RecommendationsSpace instances, the REST API generates a RecommendationsSetResource that can be serialized to different formats.

1.7.2 On The Fly recommenders Description

On The Fly Recommenders are those that need less time to carry out their algorithms, but unlike the Batch Recommenders they cannot precalculate their recommendations since some of the information is not known till runtime when the user request is inspected. We only consider those that use content-based techniques, and as such these algorithms are a variation of the Content-based recommender that we described before. They are the RO Based Recommender and the Context Based Recommender, and can be described as the following steps:

- **Step A. Creation of the description of the content of items.** This is mainly a process done in the background when different items (i.e. ROs, RO resources such as papers, workflows, presentations, etc.) are added to the system.

- **Step B. Creation of the keywords set.** The keywords set summarizes the content of the item for which we wish to find similar elements in the system. In the case of
  - **RO Based Recommender.** The recommender extracts all the items aggregated by the RO and extracts its main keywords. The keyword set is the union of all the keywords sets of the resources.
  - **Context Based Recommender.** The keyword set is composed by:
    - The keywords that the user recommendation context holds.
    - The union of all the keyword sets of each of the resources which URI is contained in the Recommendation Context of the user

- **Step C. Matching process.** We match the keyword set against the description of the content of the items. As in the case of the content-based recommender, the matching technique that we use the TF-IDF statistical measure.

1.7.2.1 On The Fly Recommenders Interactions

As in the case of Batch Recommenders, recommendations made by On The Fly Recommenders are also packed as resources. In the case of the Context Based Recommenders they are ContextualizedRecommendationsSet instances; and as we have already stated, these resources are created just in the moment when requested by the client application.
Figure 13 ContextRecommendationsSetResource context creation.

When a new ContextualizedRecommendationSetResource needs to be created, the REST API invokes the `getOnTheFlyRecommendationsSpace` method exposed by the Core class. The Core firstly retrieves the user context, which as we stated is a list of relevant resources and keywords that have already been marked as relevant by the user. Then, for each instance of the Context Based Recommender, the Core invokes its recommend method, storing all the recommendations in the same RecommendationsSpace instance. Finally this instance of RecommendationsSpace is returned, and the REST API generates a ContextualizedRecommendationSetResource that can be serialized to different formats.
Finally, when a RORecommendationSetResource needs to be created, the REST API invokes the getOnTheFlyRecommendationsSpace method exposed by the Core class. The Core firstly retrieves the RO from the Shared Model, and then, for each instance of ROBasedRecommender, the Core invokes its recommend method, storing all the recommendations in the same RecommendationsSpace instance. Finally this instance of RecommendationsSpace is returned, and the REST API generates a RORecommendationSetResource that can be serialized to different formats.

1.8 Recommender Service API

After discussing the Recommender Service architecture and design let us describe how the Recommender Service module exposes its functionality. The Recommender Service API has been implemented as a fully REST compliant interface. As such, in order to interact with the service the client first retrieves a service document that provides links that the client can navigate. This interaction goes as follows:

C: GET /recommender HTTP/1.1
C: Host: sandbox.wf4ever-project.org/epnoiServer/rest
C: Accept: application/xml

S: HTTP/1.1 200 OK
S: Content-Type: application/xml
S:
S: <recommender>

2013 © Copyright lies with the respective authors and their institutions.
The service document specifies the link relations that are central to the Recommender Service REST API so that the client application can navigate through them. Currently they are:

- **<recommendationsSet>** The set of recommendations for the user identified as user (the integer that represents the user in myExperiment). Its cardinality may be restricted up to a number (max)
  /recommender/recommendations/recommendationsSet/user/{user}/{itemType}{?max}

- **<recommendationContext>** The recommendation context must be set up in case that the user may be interested in receiving recommendations based in a group of myExperiment resources or keywords. The recommendation context is composed by the set of resources (0..N resources defined by the resource query parameter), the set of keywords (0..N keywords defined by the keyword query parameter), and the URI of the user that is associated with the context (user query param)
  /recommender/contexts/recommendationContext{?user,resource, keyword}

- **<contextualizedRecommendationsSet>** The set of contextualized recommendations for the user identified as user (user query param) of items of a type (type query param)(i.e.workflows, files, users, packs). Its cardinality may be restricted up to a number (max query param)
  /recommender/recommendations/contextualizedRecommendationsSet{?user,type, max}

- **<RORecommendationsSet>** The set of recommendations that are related with a given RO.
  /recommender/recommendations/contextualizedRecommendationsSet{?RO,type, max}

Once the client has retrieved the service document, it extracts the URI template for the Recommender Service and assembles the URI for the desired recommendations set. For example, the request for the recommendations for the myExperiment user with id 2 is the following:

C: GET /recommender/recommendations/recommendationsSet/user/2 HTTP/1.1
C: Host: sandbox.wf4ever-project.org/epnoiServer/rest
C: Accept: application/xml

S: HTTP/1.1 200 OK
The workflow entitled Get names of proteins similar to RNA binding proteins (Simple example SADI workflow) (URI:http://www.myexperiment.org/workflow.xml?id=2127) is recommended to you since you used the following tags: sadi, taverna, spreadsheet; and they partially describe its content.

The client can also assemble the URI for creating the desired recommendation context for a later use of the contextualized recommender:

C: PUT
&resource=http://www.myexperiment.org/workflow.xml?id=16
After creating the recommendation context the client can request a recommendations obtained using the provided context. Following the example, the message interchange goes as follows:

C: GET /recommender/recommendations/contextualizedRecommendationsSet?
C: user=http://www.myexperiment.org/user.xml?id=2 HTTP/1.1
C: Host: sandbox.wf4ever-project.org/epnoiServer/rest
C: Accept: application/xml

S: HTTP/1.1 200 OK
S: Content-Type: application/xml

S: <?xml version="1.0"?>
S: <recommendationsSet>
S:   <recommendation>
S:     <explanation>
S:       The workflow entitled Pathways and Gene annotations for Arabidopsis affy data (URI:http://www.myexperiment.org/workflow.xml?id=726) is recommended to you since you selected the resources ([http://www.myexperiment.org/workflow.xml?id=16, http://www.myexperiment.org/workflow.xml?id=1583]) with similar components
S:     </explanation>
S:     <itemType>item_type_workflow</itemType>
S:     <resource>http://www.myexperiment.org/workflows/726</resource>
S:     <strength>5.0</strength>
S:     <title>Pathways and Gene annotations for Arabidopsis affy data</title>
S:     <usedTechnique>technique_group_content_based</usedTechnique>
S:   </recommendation>
S:   ...
S:   <recommendation>
S:     <explanation>
S:       The workflow entitled KEGG pathways common to both QTL and microarray based investigations (URI:http://www.myexperiment.org/workflow.xml?id=13) is recommended to you since you
S:     </explanation>
S:     <itemType>item_type_workflow</itemType>
S:     <resource>http://www.myexperiment.org/workflows/13</resource>
S:     <strength>5.0</strength>
S:     <title>KEGG pathways common to both QTL and microarray based investigations</title>
S:     <usedTechnique>technique_group_content_based</usedTechnique>
S:   </recommendation>
S: </recommendationsSet>
Finally, let us specify the resources that are exposed in the Recommender Service REST API. They are:

- **RecommendationsSetResource.** A RecommendationsSetResource represents a set of recommendations for a given user, and is specified by the following schema:

  ```xml
  <recommendationsSet>
    <recommendation>
      <explanation>An user oriented description on why the recommendation is made to the user</explanation>
      <itemType>
        [item_type_workflow|item_type_file|item_type_pack|item_type_user]
      </itemType>
      <resource>The URL of the recommended item</resource>
      <strength>A real number that ranges from 0 to 5 with that represents the relevancy of the recommendation</strength>
      <title>The title or name of the recommended item</title>
      <usedTechnique>
        [technique_keyword_content_based|technique_social|technique_collaborative|technique_inferred|technique_group_content_based]
      </usedTechnique>
    </recommendation>
  </recommendationsSet>
  ...
  ```

- **ContextualizedRecommendationsSetResource.** ContextualizedRecommendationsSetResource represents a set of recommendations that are provided for the user taking into account the context defined by the user. This resource is represented following the same schema of the RecommendationsSetResource.

- **RecommendationContextResource.** The recommendationContext resource contains the group or resources and keywords that are considered in the provisioning of contextualized recommendations for a given user. They are defined by the following schema:
<recommendationContext>
  <resource>resource URI 0</resource>
  ...
  <resource>resource URI N</resource>

  <keyword>keyword 0</keyword>
  ...
  <keyword>keyword N</keyword>
  <userURI>The user associated with the context</userURI>
</recommendationContext>
3. Collaboration Spheres

1.9 Introduction
The Collaboration Spheres aim at providing a mechanism to explore, share and reuse ROs and user expertise based on the exploitation of semantic descriptions, relations and similarities between ROs and users in order to provide advanced search functionalities. This type of exploratory search is especially appropriate in domains where social aspects like collaboration and the notion of a community play a relevant role in order to incrementally expand the knowledge assets of such community. Examples of this kind of scenarios include myExperiment and the American Psychological Association\(^\text{14}\) (APA), where the Collaboration Spheres have been deployed. The Collaboration Spheres metaphor and the user interfaces implementing it leverage collaborative filtering and content-based recommendations provided by the Recommender Service, described earlier in this document. The proposed visual metaphor is based on different spheres centered around a core point, which in our case is the user. The user interface implementing this metaphor is simple and user-centric and provides the necessary connection between the underlying Wf4Ever technologies including the Recommender Service and the exploitation of RO meta-data.

1.10 The Collaboration Spheres Visual Metaphor
There are different ways in which users can express a query in order to retrieve content from a repository. Well-known approaches include, among others, faceted search\(^\text{15}\), where the user selects the most relevant features from a predefined set in order to constrain the search space, and free text query interfaces, which rely on natural language processing technologies to parse and match the query against the overall content. However, it is usually the case that users lack the precise knowledge about the exact features of the information to be retrieved or the skills required to express them in specific query formalisms.

The difficulty is considerably lower when, instead of formulating a query, users are enabled to search by example, exploiting the potential similarities between such examples and the results. We follow the same principle behind the use of examples e.g. in education, in order to facilitate the assimilation of complex concepts by students. By selecting a number of representative exemplars users can express the properties that must be observed in the expected results without explicitly or formally describing such properties. This query-by-example method relieves users from the task of formulating potentially complex queries, delegating such complexity to the underlying system.

Additionally, this approach allows users to explore the search space through the context of interest described by the aggregated properties of the selected exemplars. For example, by putting together different ROs dealing with cardiovascular diseases and diabetes and analyzing the relatedness of this context of interest against the ROs contained in the repository it is possible to establish a connection between these disorders

\(^{14}\) http://www.apa.org

\(^{15}\) Faceted Search, Morgan & Claypool, 2009

2013 © Copyright lies with the respective authors and their institutions.
and the metabolic syndrome. This kind of exploratory search is specially aimed towards gaining new insights on existing information and discovering relations between them that were not previously explicit.

Recommendation technologies allow establishing such similarities between a context of interest and the search space. However, in order to be effective these functionalities need to be provided to users in a way that simplifies interaction, especially when it comes to creating the context of interest and visualizing the results of the exploratory search.

Our visualization metaphor and the prototype interface implementing it make use of concentric spheres that represent different types of similarity metrics between the context of interest, expressed by the user as a collection of ROs and other users that the user finds relevant for a particular purpose, and the results obtained by the recommenders. The distance between the center, i.e. the user and the context of interest, and the two external spheres, where recommendation results are displayed, provides a notion of confidence about the recommendations. The closer to the center, the more specific the recommendation result will be with respect to the user and the current context of interest.

The innermost of the two external spheres where search results are shown is populated through model-based recommendation algorithms that determine the similarities between the context of interest and potential results. This type of recommendation exploits information obtained from RO meta-data and user profiles. The results of model-based recommendation are complemented with the content of the outmost sphere, which is populated with results produced by means of collaborative filtering algorithms that take into account the historical selections and past preferences of the user. Thus, this outmost sphere is not specific to the current context of interest but on the contrary reflects the preferences of a particular user throughout previous interactions.

### 1.11 The User Interface – Core Concepts

The user interface associated to the Collaboration Spheres visual metaphor contains four main spheres:

- **The User.** It displays the active user.

- **The Inner Sphere (context of interest).** It represents the context of interest. This circle contains the users and ROs that are selected or pre-defined by the active user. In order to create a context of interest, the inner sphere is populated by drag-and-dropping relevant ROs and users from lists ranked by relevance.

- **The Intermediate Sphere (model-based recommendation).** It represents information associated to the recommendation obtained by using the context of interest as input for the recommendation techniques. The ROs and members of social network displayed in this sphere correspond to suitable items according to the context of interest defined in the previous sphere. It follows an inside-outside criterion where the inner part flows towards the outside. At this stage the recommendation obtained by this layer uses a content-based recommendation approach based on tags and annotations. This is a social-oriented approach and it is based on the fact that people are many times more reliable than search engines ("I trust my friends more than I trust strangers.").
• **The Outer Sphere (user-based recommendation).** It contains recommended ROs and users based on past user actions rather than in the information explicitly defined in the context of interest. In order to populate this sphere we also rely on predictive models that provide the user with new possible interests.

The collaboration spheres interface is agnostic from the different similarities calculated by the underlying recommenders. Currently we support the following three main types of similarities:

- **RO vs. RO.** It represents ROs that share common properties. This relation is evaluated by analyzing the similar or identical tags described in the ROs. Therefore it is classified as **content-based** type.

- **User vs. User.** It represents users that share common interests, and is aimed towards supporting information exchange about the domain and the activities being carried out. This relation is evaluated by analyzing the similarity between tags contained in the user profiles and associated information. Therefore it is classified as **content-based**.

- **RO vs. User.** It represents the functionalities that are shared with user interest or vice versa. This relation is evaluated by analyzing the similar or identical tags described in the user profile and RO tags (**content-based**), and by analyzing historical/related use of ROs by a user and predicting new possible RO’s likes (**collaborative filtering** type).

### 1.12 Implementation

Figure 15 shows the API and sequence diagram of the Collaboration spheres. The diagram illustrates how the interface and the underlying recommendation and Wf4Ever RO management infrastructure interplay. The figure shows how similar collaborators and ROs to those introduced in the context of interest circle are obtained via SPARQL query to the source repository e.g. myExperiment. With the same user ID, the Collaboration Spheres service retrieves the General Recommendation associated to the user. At that time, the user can see the content of the intermediate sphere. After this, every time the user adds an RO or a collaborator dragging it to the context of interest, the service retrieves a Contextual Recommendation sending the RO URI or the collaborator URI as parameters to the Recommendation Service. Finally the spheres are updated based on this user-customized information and a contextual recommendation is produced.

**Component Card**

<table>
<thead>
<tr>
<th>Client Name</th>
<th>Collaboration Spheres</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A graphical user interface for the visualization of correlation between similar objects (e.g., users, Research Objects) based on collaborative filtering and versatile keyword content-based recommendations. It implements a visual metaphor based on spheres, which uses concentric circles, where the similarity is represented using the distance from the center of the sphere to the place where the object is shown, and separated circles, where different type of information or ranges can be displayed by representing different separated circles. The key features are:</td>
</tr>
<tr>
<td></td>
<td>• Creation of a three-layer representation for social data and recommendation</td>
</tr>
</tbody>
</table>
A visualization based on a user-centric three layers metaphor, with (i) customizable direct links (inner circle), (ii) recommendation based on the objects allocated in the inner circle (intermediate circle), and (iii) recommendations based on historic user profile (outer circle).

• Links recommendation work and visualization (pursuing a virus network effect)
• Use data from myExperiment

URI  
http://sandbox wf4ever-project.org/Collab/circles.html

Source Code Repository  
https://github.com/wf4ever/Collaboration-spheres

Interface  
Graphical User Interface (GUI)

The collaboration spheres client uses the recommender service and imports data from myExperiment.

Use of APIs

The Recommender API defines a service for providing recommendations of resources for a user. The types of resources that are recommended depend on the domain for which the API is used; for Wf4Ever these are users, Research Objects and their aggregated resources (e.g., scientific workflows, dataset, etc.).

When using the API, the client may define the maximum number of recommendations and narrow the type of resources recommended.

Recommender API operations:
• The `GetGeneralRecommendation` method gets the historical recommendation for a given user and its profile.
• The `GetContextualRecommendation` method is in charge of creating a context of resources in the recommender service and retrieving a customized recommendation.

![Collaboration Spheres - Sequence Diagram and API](image)

**Figure 15: Collaboration Spheres - Sequence Diagram and API**

1.13 Deployment

1.13.1 myExperiment

The previously introduced approach has been extended with information access tools and methods, as well as design decisions aimed towards improving usability, that support the selection of ROs for the construction of the context of interest and the visualization of the resulting recommendations. Such holistic collaboration spheres site has been integrated within the myExperiment Alpha platform. See Figure 16 including the different parts contained in the interface.

• **Four-layer representation.** It is the core of the visualization and it follows the description of the collaborations spheres visual metaphor. ROs and users are identified by circles and squares, respectively. The different layers are presented by using different colors.

• **List of users ordered by.** It provides a list of the users ordered by their relevance for the current active user.
• **List of ROs ordered by.** It provides a list of the ROs ordered by its relevance for the current active user.

• **Information of the selected item.** It provides information related with the selected item from the four-layer representation (e.g. workflow representation).

• **Graph of relations between items:** it provides a graph representation of the relations between users and ROs.

The data sources involved are accessed through the myExperiment\(^{16}\) API and the Recommender Service API.

---

**Four layer representation**

- List of users ordered by...
- List of ROs ordered by...
- Information about the selected item
- Graph of relations between items

*API interface definition: [www.wf4ever-project.org/wiki/display/docs/CollaborationSpheres#CollaborationSpheres-WebServices](http://www.wf4ever-project.org/wiki/display/docs/CollaborationSpheres#CollaborationSpheres-WebServices)*

---

**1.13.2 The American Psychological Association (APA)**

A slightly different approach has been followed in the case of APA, the major US publisher in the Psychology domain. The main interest of APA is to provide their users with intuitive means to identify relevant authors and articles not only in terms of the topics contained but mainly through their potential relations with other articles and authors. This deployment exploits and is built on top of APA’s documental base, which covers thousands of publications and their corresponding authors. This work focuses on a subset of APA’s articles...

related to Posttraumatic Stress Disorder (PTSD). This case is especially relevant since it is the first deployment of this technology outside the Wf4Ever consortium. In this case, we use the same fundamental visual metaphor as with myExperiment, the main differences being:

- The main piece of information to be considered is no longer an RO, whose main entity is the workflow, but the article, extended with additional metadata. This illustrates the exploitation of non-workflow ROs in industrial domains.
- An optimized and more effective method to create the context of interest.

Due to the large number of users and articles in APA, the creation of the context of interest based on those items needs to be more precise in its presentation to the user. Otherwise, users could have difficulties to select relevant authors and articles. On the right hand side of the interface we have two columns with authors and articles respectively. In the case of authors, we have classified them in three main categories: co-authors, authors in the user's citation network and others. Co-authors are those authors with whom the user has published some article, authors in the citation network are those who are not co-authors but who have been cited by the user or by other author connected to the user through a succession of cites, and other authors are all those authors who are not connected with the user. The rightmost column is analogous for the case of articles. The lists of authors and articles can be ranked following different criteria which make it easier for the user to find them, including alphabetical order, proximity in the citation network, or a mix of both.

The Collaboration Spheres deployment in the APA case is illustrated in Figure 17. In addition to the changes related to the method for creation of the context of interest introduced above, the main differences in the interface include:

- **A Tag Cloud.** This tag cloud shows, in a summarized and visual way, the set of tags and concepts associated to the aggregated users and articles contained in the Collaboration Spheres for a particular context of interest. The user is thus enabled to have a better grasp of the main objective of the recommendations he is getting in a glance.

- **Contextual Item Information.** After clicking on any of the items (an author or an article), a brief summary of information about such item, e.g. other articles by the same author and the abstract of the article, appear in this box. By means of a lightweight integration with Twitter, we also provide access to Twits related to the article or published by the authors elected.

The data sources involved come from parsed APA publications (in formats ranging from PDF to xml) in APA’s PsycNET\(^\text{17}\) and user profiles that are available at APA VIVO\(^\text{18}\), a research discovery tool looking forward collaborations among scientists. This work evolves previous deployments e.g. in myExperiment,

---

\(^{17}\) [http://psycnet.apa.org](http://psycnet.apa.org)

\(^{18}\) [https://vivo.apa.org](https://vivo.apa.org)

2013 © Copyright lies with the respective authors and their institutions.
which are planned to be updated with the latest refinements. Deployment of the Collaboration Spheres is also planned for the RO portal. Four Layer Representation

Authors

Articles in three levels

Authors in three levels

Articles

Tag Cloud

Item Information

Figure 17 Collaboration Spheres for APA demo

19 http://sandbox.wf4ever-project.org/portal/home
4. Summary

In this document we have portrayed the Wf4Ever Phase II status of the WP3 software and specifications related with the maximization of the reuse and sharing of workflow-centric ROs.

We have depicted the lifecycle of an RO, how it transitions since its conception until its conclusion. We have also described the roevo ontology that enables the representation of these different stages of the ROs lifecycle, along with their dependencies.

Regarding the software, we have described:

- The software that supports the proposed RO evolution model. That includes an evolution API for the RO model, and software for supporting the RO evolution visualization in the RO portal.
- The Recommender Service, which suggests ROs, scientific content and data, publications that might be relevant to Wf4Ever users.
- The Collaboration Spheres prototype that practically depicts how the use of such mechanism introduces advanced exploratory search mechanism usable in different scenarios.
5. References


