Wf4Ever: Advanced Workflow Preservation Technologies for Enhanced Science

STREP FP7-ICT-2007-6 270192

Objective: ICT-2009.4.1b — "Advanced preservation scenarios"

D2.2v2 Design, implementation and deployment of workflow lifecycle management components - Phase II

Deliverable Co-ordinator: **Khalid Belhajjame**

Deliverable Co-ordinating Institution: **University of Manchester**

Other Authors: **Daniel Garijo, Graham Klyne**

This deliverable describes the second phase of delivery of workflow lifecycle management components. It includes a description of the Research Object Model, which facilitates interoperation between components; the RO Manager command line tool; the Research Object Digital Library; the RO-enabled myExperiment; and a definition of models for workflow abstraction and indexation.

Document Identifier:	Wf4Ever/2013/D2.2v2/0.1	Date due:	June 30, 2013
Class Deliverable:	Wf4Ever FP7-ICT-2007-6 270192	Submission date:	June 30, 2013
Project start date	December 1, 2010	Version:	0.1
Project duration:	3 years	State:	Draft
		Distribution:	Public

Wf4Ever Consortium

This document is 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:

Intelligent Software Components S.A. (ISOCO) – Coordinator Edificio Testa, Avda. del Partenón 16-18, 1^o, 7^a Campo de las Naciones, 28042 Madrid Spain Contact person: Jose Manuel Gómez Pérez E-mail address: jmgomez@isoco.com	**University of Manchester (UNIMAN)** School of Computer Science Oxford Road, Manchester M13 9PL United Kingdom Contact person: Carole Goble E-mail address: carole.goble@manchester.ac.uk
Universidad Politécnica de Madrid (UPM) Departamento de Inteligencia Artificial, Facultad de Informática. 28660 Boadilla del Monte. Madrid Spain Contact person: Oscar Corcho E-mail address: ocorcho@fi.upm.es	**Instytut Chemii Bioorganicznej PAN - Poznan Supercomputing and Netowrking Center (PSNC)** Network Services Department Ul Z. Noskowskiego 12-14 61704 Poznań Poland Contact person: Raul Palma E-mail address: rpalma@man.poznan.pl
University of Oxford (OXF) Department of Zoology South Parks Road, Oxford OX1 3PS United Kingdom Contact person: Jun Zhao, David De Roure E-mail address: jun.zhao@zoo.ox.ac.uk david.deroure@oerc.ox.ac.uk	**Instituto de Astrofísica de Andalucía (IAA)** Dpto. Astronomía Extragaláctica. Glorieta de la Astronomía s/n, 18008 Granada Spain Contact person: Lourdes Verdes-Montenegro E-mail address: lourdes@iaa.es
Leiden University Medical Centre (LUMC) Department of Human Genetics Albinusdreef 2, 2333 ZA Leiden The Netherlands Contact person: Marco Roos E-mail address: M.Roos1@uva.nl	

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

- iSOCO
- OXF
- PSNC
- UNIMAN
- UPM

Change Log

Version	Date	Amended by	Changes
0.1	01-06-2013	Khalid Belhajjame	Initial outline
0.2	09-06-2013	Khalid Belhajjame	Initial draft of Section 2 on the RO model
0.3	12-06-2013	Graham Klyne	Added the RO manager section
0.4	14-06-2013	Daniel Garijo	Added the workflow abstraction section
0.5	14-06-2013	Khalid Belhajjame	Added the introduction and a first draft of the myExperiment section
0.6	14-06-2013	Khalid Belhajjame	Revised all sections and added the summary section
0.7	21-06-2013	Khalid Belhajjame	Addressed QA comments received from Oscar Corcho on all sections, except Sections 4 and 7
0.8	21-06-2013	Raul Palma	Research Object Digital Library section
0.8.1	21-06-2013	Piotr Hołubowicz	Research Object Digital Library UML diagrams and alignment
0.8.2	24-06-2013	Raul Palma	Research Object Digital Library section references and fixes

Executive Summary

This deliverable describes the second phase of delivery of workflow lifecycle management components. These components are focused around the Wf4Ever Research Object Model (RO Model), which provides descriptions of workflow-centric ROs – aggregations of content. This model is used to structure and describe ROs which are then stored and manipulated by the components of the Wf4Ever Toolkit.

The RO Model provides a framework for describing aggregations of content along with annotations of the aggregated resources, a vocabulary for describing workflows, and a vocabulary for describing provenance. The model did not undergo any major changes in the in the last year, which is a good sign as it suggests that the model is mature enough and captures user requirements adequately. We provide here a description of the RO model. We also present the components developed for creating and managing Research Objects: the RO Manager – the Research Object Digital Library. These components and services are also discussed in D1.2v3 (Wf4Ever Sandbox – Phase II), D1.3v2 (Wf4Ever Architecture – Phase II) and D1.4v2 (Reference Wf4Ever Implementation – Phase II).

One of the main developments in the last year consist in incorporating research objects within the myExperiment environment to allow scientists who already use myExperiment to create, share and reuse research objects. We discuss the efforts that went into this task, and show how myExperiment is using Research Object Digital Library as a back-end for storing and archiving Research Objects.

We present advanced management functions that we developed for abstracting and indexing workflows, with the aim of supporting the discovery and reuse of workflows. We present an ontology that we developed for abstracting workflows in terms of motifs that characterize data manipulation and transformation patterns, which we term motifs. We also report on a solution that we developed for indexing workflows based on the services (processes) that they use.

This deliverable should be read in tandem with D1.3v2 (Wf4Ever Architecture – Phase II), D1.4v2 (Reference Wf4Ever Implementation – Phase II), D1.2v3 (Wf4Ever Sandbox – Phase III), D3.2v2 (Design, implementation and deployment of Workflow Evolution, Sharing and Collaboration components – Phase II) and D4.2v2 (Design, implementation and deployment of Workflow Integrity and Authenticity Maintenance components – Phase II) in order to provide a complete picture of the state of the Wf4Ever Phase II components.

Contents

List of Figures

List of Ontologies

1. `RO` ontology: `http://purl.org/wf4ever/ro#`
2. `Wfdesc` ontology: `http://purl.org/wf4ever/wfdesc#`
3. `Wfprov` ontology: `http://purl.org/wf4ever/wfprov#`
4. `ROEvo` ontology: `http://purl.org/wf4ever/roevo#`

1 Introduction

This deliverable describes Phase II of the design, implementation and deployment of the Wf4Ever components that will support workflow lifecycle management. The document should be read in tandem with other Month 32 deliverables, in particular D3.2v2 (Design, implementation and deployment of Workflow Evolution, Sharing and Collaboration components – Phase II) [**?**] and D4.2v2 (Design, implementation and deployment of Workflow Integrity and Authenticity Maintenance components – Phase II) [**?**] which address complementary aspects of the overall wf4ver architecture and components.

According to the Description of Work, *This prototype will include the following functionalities: new versions of the Research Object model and ontology network, advanced management functions (filtering, clustering, etc.), playback functionalities for reproducibility, and workflow classification, indexing and explanation techniques.*.

These requirements are addressed in the following way:

Section 2 presents the Research Object Model defined within Wf4Ever. Specifically, we present a family of ontologies that we developed for specifying Research Objects and their associated resources, i.e., workflow, workfllow runs, etc.

Sections 3.1, 3.2 and 3.3 present the tools that we developed for assisting users in creating and managing Research Objects. Section 3.1 presents the Research Object Manager (RO Manager), a command line tool for creating, displaying and manipulating Research Objects. Section 3.2 presents the Research Object Digital Library (RODL), which acts as a full-fledged back-end not only for scientists but also for librarians. Finally, Section 3.3 shows how the myExperiment virtual environment [**?**], was extended to allow end-users, who are not necessarily information technology experts, to create, share, publish and curate Research Objects.

Section 4 presents the motif ontology that we developed for abstracting scientific workflows, and illustrates how it has been used to document workflows, while Section 5 presents a solution that we developed for indexing workflows based on the processes (steps) they are composed of, with the purpose of assisting users in discovering workflows that are of interest to them.

2 The Research Object Model

The design of the Research Object model was informed by a systematic analysis of requirements expressed by scientists from the life sciences and astronomy fields. In the last year, the Research Object model didn't undergo any major changes, which is a good sign as it means that the modele is mature enough to capture the requirements of the user. Figure 1 describes the abstract model of the Research Object model, distinguishing between core and extended requirements. There are three core requirements that have been identified, namely a mechanism for uniquely identifying Research Objects, a means for aggregating resources within a Research Object, and the ability to annotate the Research Object, its constituent resources and their relationships. Based on the core requirements, the extended requirements highlight the need for specifying workflows (experiments), provenance traces of their executions, the evolution of a Research object over time, as well as mechanisms for citing Research Objects, expressing their dependencies, etc.

We have realized the Research Object abstract model illustrated in Figure 1 in the form of a family of ontologies that are illustrated in Figure 2, which we will present in the rest of this section. It is worth noting that some of the vocabularies, e.g., ORE[1] and OA [**?**], are existing vocabularies that we built on to specify our ontologies.

2.1 RO core ontology

The Core RO Ontology provides the minimum terms that are essential to the specification of research objects. Specifically, it caters for two essential requirements by providing a container structure that can be used by the

[1]`www.openarchives.org/ore`

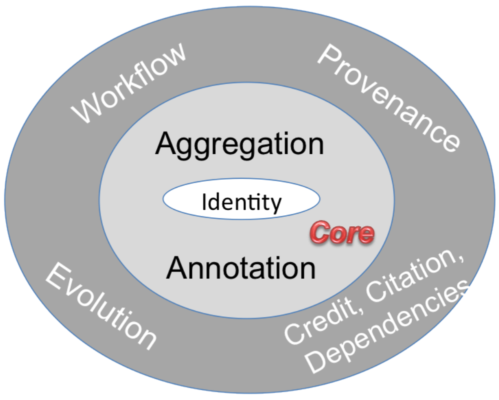


Figure 1: Research Objects: Abstract model.

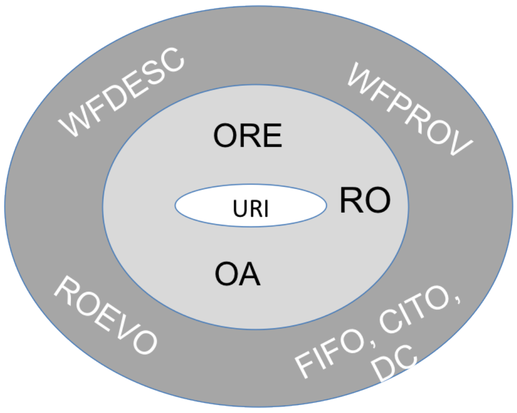


Figure 2: Research Objects: Concrete model.

scientists to bundle the resources and material relevant for their investigation, and by enabling annotations of such a container, its resources, as well as the relationships between resources thereby making the research object interpretable and reusable.

To cater for the specification of aggregation structures, we built the Research Object Core Ontology upon the popular ORE vocabulary. ORE defines standards for the description and exchange of aggregations of Web resources. Figure 3 illustrates the main terms that constitute the Research Object Core Ontology, which we describe in what follows.

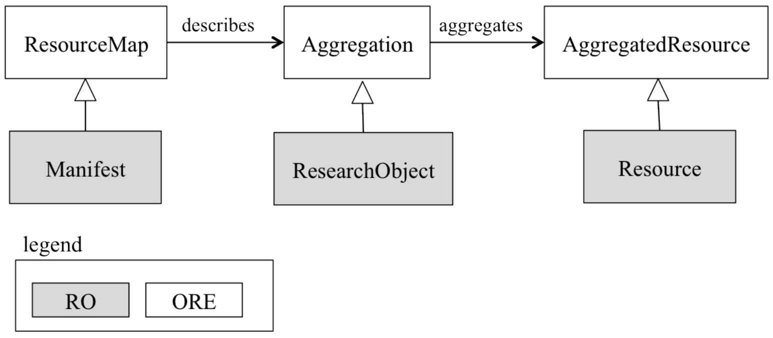


Figure 3: RO as an ORE aggregation.

- `ro:ResearchObject`[2], represents an aggregation of resources. It is a sub-class of `ore:Aggregation` and acts as an entry point to the research object.

[2]The namespace of the Research Object Core Ontology `ro` is `http://purl.org/net/wf4ever/ro#`

- ro:Resource, represents a resource that can be aggregated within a research object and is a sub-class of ore:AggregatedResource. A resource can be a Dataset, Paper, Software or Annotation. Typically, a ro:ResearchObject aggregates multiple ro:Resource, and this relationship is specified using the property ore:aggregates.
- ro:Manifest, a sub-class of ore:ResourceMap, represents a resource that is used to describe a ro:ResearchObject. It plays a similar role to the manifest in a JAR or a ZIP file, and is primarily used to list the resources that are aggregated within the research object.

The second core requirement that, the Research Object Core Ontology caters for, is the descriptions of the research object and its elements. We chose the Annotation Ontology (AO) release 2.0b2 [**?**].To annotate research objects, we make use of the following three Annotation Ontology terms ao:Annotation[3], which represents the annotation itself; ao:Target, which is used to specify the ro:Resource(s) or ro:ResearchObject(s) subject to annotation; and ao:Body, which comprises a description of the target. In the case of research objects, we use annotations as a mean for decorating a resource (or a set of resources) with metadata information. The body is specified in the form of a set of RDF statements, which can be used to, e.g., specify the date of creation of the target or its relationship with other resources or research objects. Also, annotations can be provided for human consumption (e.g. a description of a hypothesis that is tested by a workflow-based experiment), or for machine consumption (e.g. a structured description of the provenance of results generated by a workflow run). Both kinds of annotations are accommodated using Annotation Ontology structures.

2.2 RO Extension Ontologies

We present in this section two extensions to the core Research Object ontology. The first specializes the kinds of resources that the research object can aggregate. In particular, we present extensions to specify method and experiments and the traces of their executions. The second kind of extension shows how specific metadata information, specifying the evolution of the research object over time, can be specified by specializing the Research Object core ontology.

Specifying Workflows To describe workflow research objects the workflow description vocabulary *wfdesc*[4] defines several specific resources that are involved in a workflow specification. The choice of these resources was performed by examining the commonalities between major data driven workflows, namely Taverna[5], Wings[6] and Galaxy[7], to cite a few.

Figure 4 illustrates the terms that compose the *wfdesc* ontology. Using such ontology, a workflow is described using the following three main terms:

- wfdesc:Workflow refers to a network in which the nodes are processes and the edges represent data links. It is defined as a subclass of the *Plan* concept from the PROV-O ontology, which represents a set of actions or steps intended by one or more agents to achieve some goals [**?**].
- wfdesc:Process is used to describe a class of actions that when enacted give rise to process runs. Processes specify the software component (e.g., web service) responsible for undertaking those actions.
- wfdesc:DataLink is used to encode the data dependencies between the processes that constitute a workflow. Specifically, a data link connects the output of a given process to the input of another process, specifying that the artifacts produced by the former are used to feed the latter.

[3]The namespace of ao is http://purl.org/ao/
[4]The name space of *wfdesc* is http://purl.org/wf4ever/wfdesc#.
[5]http://www.taverna.org.uk
[6]http://http://wings-workflows.org
[7]http://galaxyproject.org

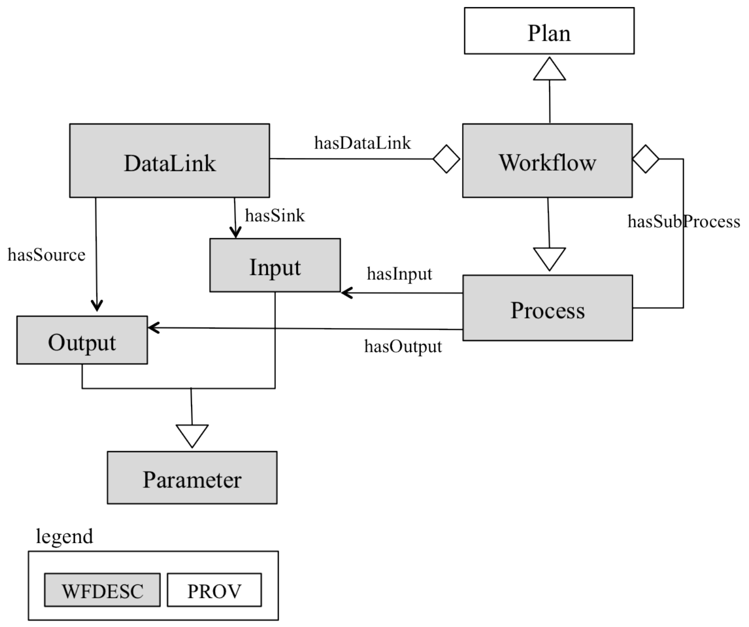


Figure 4: The *wfdesc* ontology.

Describing Experimental Provenance using the *wfprov* Vocabulary The *wfprov* ontology is used to describe the provenance traces obtained by enacting workflows. It is defined as an extension to the ongoing W3C PROV standard ontology - PROV-O[8].

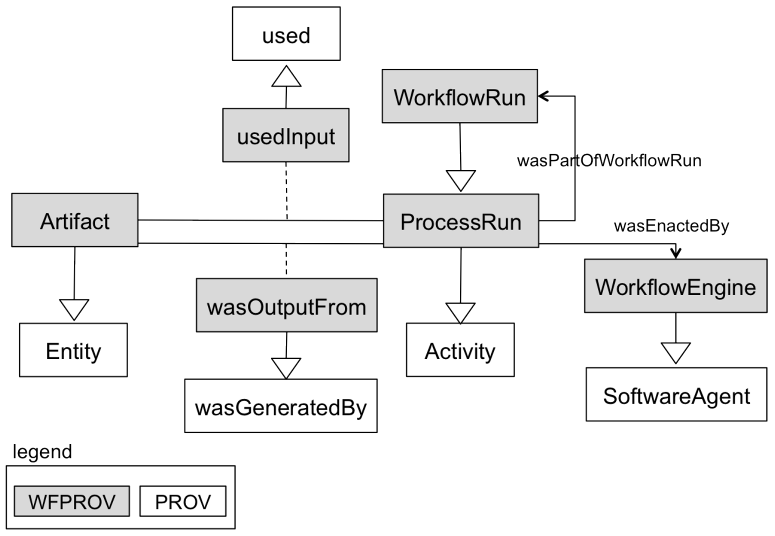


Figure 5: The *wfprov* ontology.

Figure 5 illustrates the structure of the *wfprov* ontology and its alignments with the W3C PROV-O ontology. A a workflow run (`wfprov:WorkflowRun`) represents the enactment of a given workflow. It is composed of a set of process runs (`wfprov:ProcessRun`), each representing the enactment of a process. A process run may use some artifacts (`wfprov:Artifact`) as input and generate others as output. A process run is enacted by a workflow engine (`wfprov:WorkflowEngine`), which can be seen as a PROV software agent.

By chaining the usage and generation of artifact together, the *wfprov* ontology allows scientists to trace the lineage of workflow results. For example the user can identify the input artifacts that were used to feed the wokflow run (as a whole) to obtain a given output that was generated by the workflow run.

[8]Note that the *wfprov* is reported in the W3C PROV Working Group implementation report.

Tracking Research Object Evolution using the *roevo* Vocabulary The *roevo* ontology is another extension to the minimal core ontology for describing an important aspect of research objects, its life cycle. To track the life cycle of a research object, we need to describe its changes at different levels of granularity, about the research object as a whole and about the individual resources. Also, we want to provide sufficient details to track the changes in order to roll back to a particular version or to quality control changes. Therefore, we need to describe when the change took place, who performed the change, and dependency relationships between the changes. Change is closely related to the provenance of a particular version of a research object or a resource. A study of the latest PROV-O ontology shows that it indeed provides all the foundational information elements for us to build the evolution ontology.

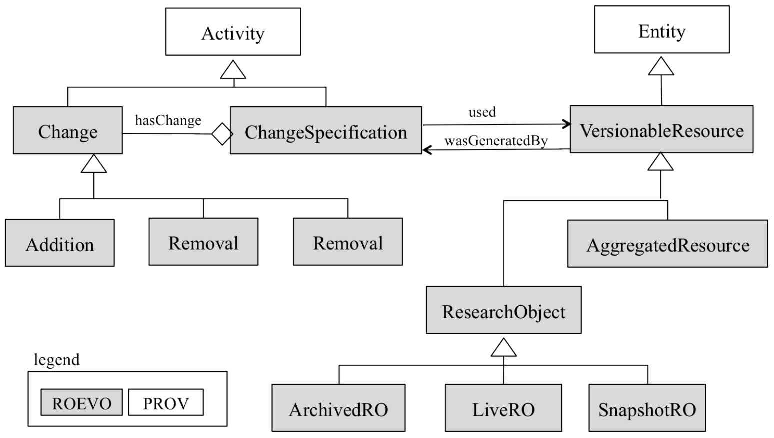


Figure 6: The *roevo* ontology extending PROV-O core terms.

Figure 6 illustrates the core concepts of this ontology and how it extends the PROV-O:

- To capture different status of a research object we create three sub-classes of `ro:ResearchObject`: the `roevo:LiveRO` is a research object to capture research findings during a live investigation and it can changed, and it can either archived or snapshotted. The `roevo:ArchivedRO` can be regarded as a production research object to be preserved and archived, such as one describing findings published in an article, and it can no longer be changed; the `roevo:SnapshotRO` represents a live Research Object at a particular time.
- Both a snapshot of a live Research Object and an archived Research Object can be regarded as a versioned Research Object, i.e. a `roevo:VersionableResource`, Because it is a sub-class of `prov:Entity`, we can reuse PROV-O properties to describe the provenance or changes of this entity, such as pointing to the activity leading to any of its changes, the source research object that it was derived from, and the agent involved in its change.
- A change is a `prov:Activity`, which means that it has a start time, an end time, an input entity and a resulting entity. Also a change leading to a new Research Object can constitute a series of changes. Therefore, we have a composite `roevo:ChangeSpecification` activity, which has a number of unit `roevo:Changes`. A unit change can be adding, removing or modifying a resource or a research object. But these different changes share the same pattern of taking an input entity and producing an output entity, which can all be nicely covered by properties from PROV-O.

As well as the above vocabularies, the Research Object model makes use of existing vocabularies, in particular, FOF[9], DCTerms[10], CITO[11], and SCIOC[12] to provide Research Objects designers with the means

[9]`http://xmlns.com/foaf/spec/`
[10]`http://dublincore.org/documents/dcmi-terms/`
[11]`http://vocab.ox.ac.uk/cito`
[12]`http://sioc-project.org/ontology`

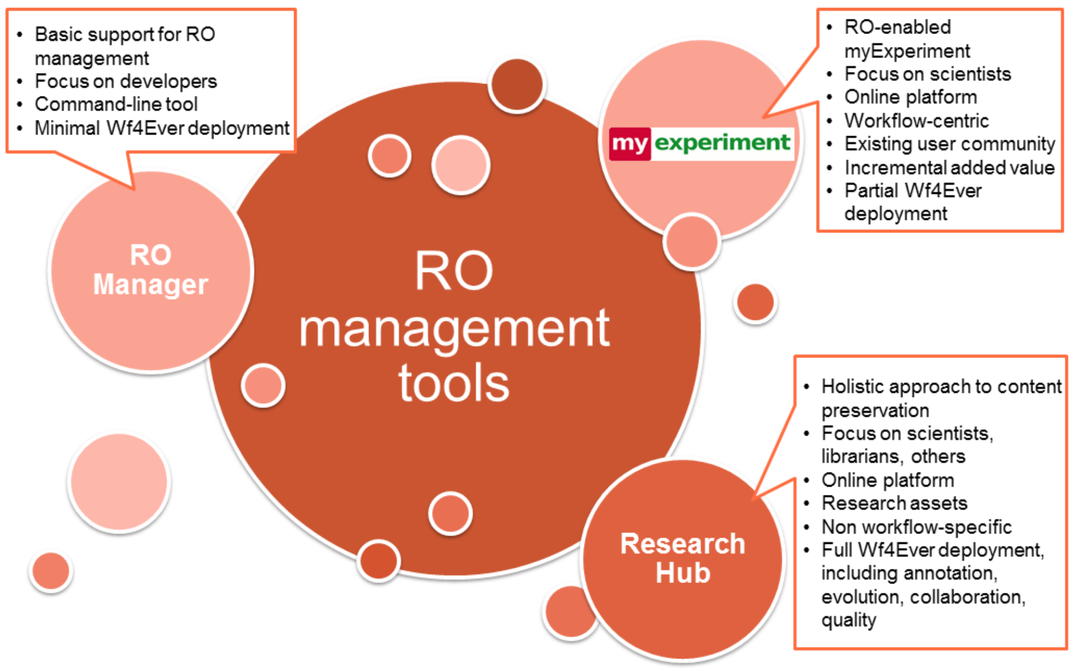

• Basic support for RO management
• Focus on developers
• Command-line tool
• Minimal Wf4Ever deployment
RO Manager
RO management tools
my experiment
• RO-enabled myExperiment
• Focus on scientists
• Online platform
• Workflow-centric
• Existing user community
• Incremental added value
• Partial Wf4Ever deployment
Research Hub
• Holistic approach to content preservation
• Focus on scientists, librarians, others
• Online platform
• Research assets
• Non workflow-specific
• Full Wf4Ever deployment, including annotation, evolution, collaboration, quality

Figure 7: Research Object management tools.

to expresse aspects such as the people who were involved in the creation of a Research Object, its citation, as well as dependencies that the Research Object may have. For instance, we make use of the term `dc:requires` to specify that a the execution of a workflow requires other resources, e.g., plugins, credential, or specific execution environment.

3 Research Object Management Tools

3.1 Research Object Manager

The Research Object Manager (RO Manager) is a command line tool for creating, displaying and manipulating Research Objects. The RO Manager is complementary to RODL (see Sectio 3.2), in that it is primarily designed to support a user working with ROs in the host computer's local file system, with the intention being that the RODL and RO Manager can exchange ROs between them, using of the shared RO model and vocabularies. The RO Manager code base also includes the checklist evaluation functionality, described in D4.2 [**?**], which can be invoked using a command line or REST web interface.

Experience has shown that a simple command-line tool can provide developers and users with early access to functionality, and provide an opportunity to gather additional user feedback and requirements. RO Manager has also been used in conjunction with built-in operating system functionality for scripting prototype tool chains for more complex operations involving Research Objects.

The RO Manager allows users and developers to:

- Create local ROs;
- Add resources to an RO;
- Add annotations to an RO;

- Read and write ROs to the RODL;
- Perform checklist evaluation of an RO;
- Obtain a raw dump of Research Object metadata.

To illustrate how the user can interact with the RO manager to manipulate research objects. Figure 8 shows interactions for three typical RO Manager operations, `ro create`, `ro add` and `ro annotate`, which exemplify typical local RO management operations.

The four interacting elements presented are the user-issued command (`/user`), the RO Manager program (`/RO_Manager`'), an internal RO metadata object (`/ro_metadata`') that manages the RO aggregation and annotation metadata, and the local file system (`/file_system`) where ROs are persistently stored and managed.

From this, it can be seen that:

- The `ro create` command initializes an RO structure by interacting directly with the file system.
- The `ro add` command uses the RO URI to initialize an `ro\_metadata` object, and calls its `addAggregatedResources()` method to incorporate one or more files into the RO aggregation. The `ro\_metadata` object updates the RO metadata structures in the file system through a series or read and write operations.
- The `ro annotate`' command similarly uses the RO URI to initialize an `ro\_metadata` object, and reads the existing annotations from disk. New annotations may be supplied as an attribute/value or attribute/link pair in which a case a new annotation graph is created in the file system. Otherwise the new annotation may already exist as a graph. In either case, the local copy of the RO manifest is updated to record the new annotation. The annotation may be applied to multiple resources in the RO. Eventually, the updated manifest is written to the file system by the `ro\_metadata` object.

The RO manager is documented in a user guide, that is available online[13]. An FAQ describing how to deal with various common operations using RO Manager is also accessible online [14].

The RO Manager is implemented in Python, and is available as an installable package through the Python Package Index (PyPI) [15]. The source code is maintained in the Wf4ever Github repository[16]. The RO Manager is heavily dependent on RDFLib[17], which provides RDF parsing, formatting and SPARQL Query capabilities. The RO Web service uses the Pyramid[18] web framework, and uritemplate[19] for RFC 6570[20] template expansion.

3.2 Research Object Digital Library

The foundational service to preserve workflow-centric research objects is the Research Object Digital Library (RODL), which realizes the Storage and Lifecycle functionalities prescribed by Wf4Ever Architecture [**?**]. RODL is a software system which collects, manages and preserves aggregations of scientific workflows and related objects and annotations, packed into research objects.

[13] `http://wf4ever.github.io/ro-manager/doc/RO-manager.html`
[14] `http://www.wf4ever-project.org/wiki/display/docs/RO+Manager+FAQ`
[15] `https://pypi.python.org/pypi/ro-manager`
[16] `https://github.com/wf4ever/ro-manager`
[17] https://github.com/RDFLib
[18] http://docs.pylonsproject.org/projects/pyramid/
[19] http://code.google.com/p/uri-templates/
[20] http://tools.ietf.org/html/rfc6570

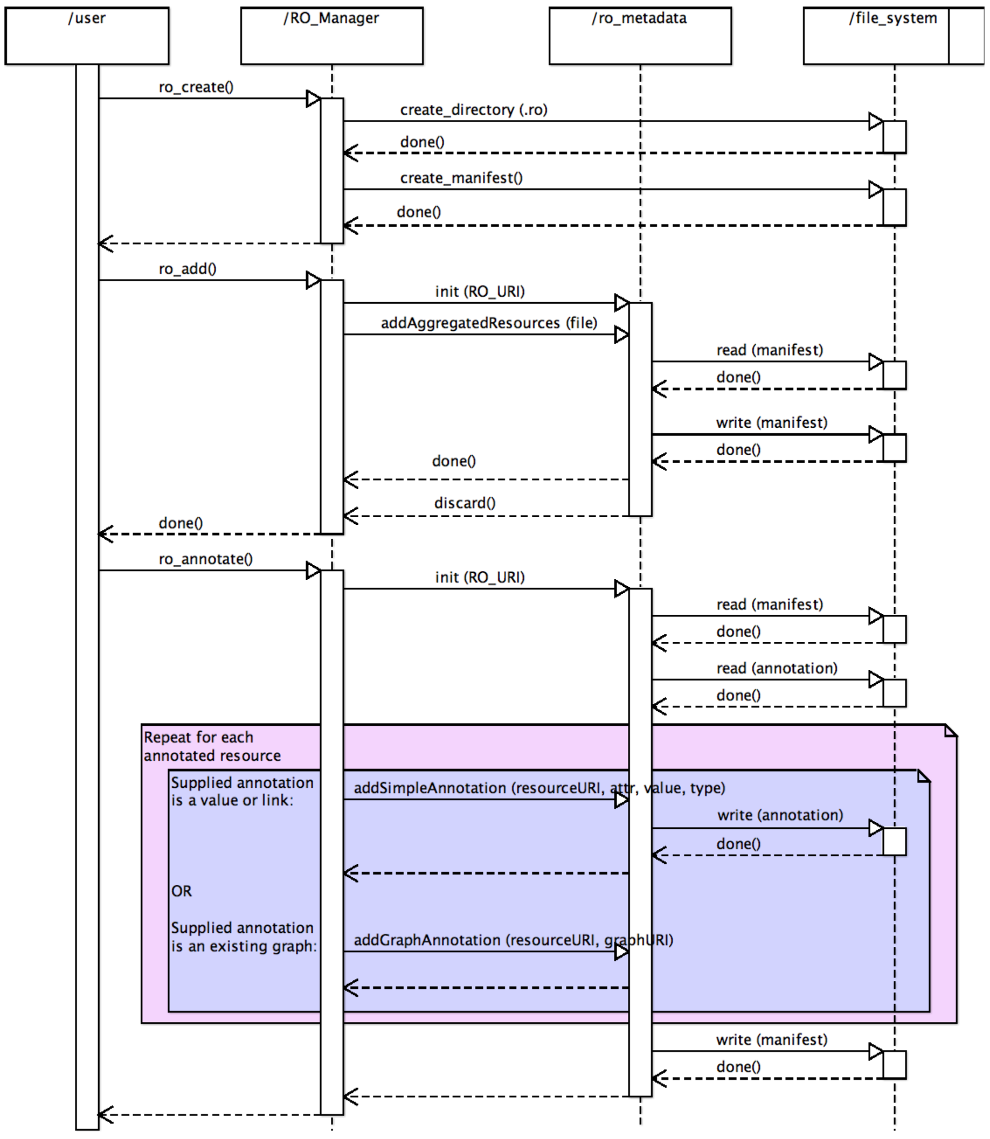


Figure 8: RO Manager sequence diagram illustrating interactions with the user.

3.2.1 The interfaces

The main interface of RODL is a set of REST APIs, among which the two primary ones are the RO API [**?**] and the RO Evolution API [**?**].

The RO API, also called the RO Storage and Retrieval API, defines the formats and links used to create and maintain research objects in the digital library. It is aligned with the RO model that is used to define research objects, and so it recognizes concepts such as aggregations, annotations and folders. The RO model ontology [**?**] is used to specify relations between different resources. Given that the semantic metadata are an important component of a research object, the RODL supports content negotiation for the metadata resources, including formats such as RDF/XML, Turtle and TriG.

The RO Evolution API defines the formats and links used to change the lifecycle stage of a research object, most importantly to create an immutable snapshot or archive from a mutable live research object, as well as to retrieve the evolution provenance of a research object. The API follows the RO evolution model [**?**], which is most visible in the evolution metadata that are generated for each state transition.

Additionally, RODL provides a SPARQL endpoint that allows performing SPARQL queries over HTTP to the metadata of all stored research objects. It also implements the Notification API [**?**], which defines links used to retrieve Atom feeds with notifications of events about any research object. For searching the contents of research objects a Solr REST API and the OpenSearch APIs are provided. Finally, RODL implements a custom User Management API [**?**] for registering users and generating OAuth 2 access tokens, providing the option of extending it with an access control layer in the future.

3.2.2 The implementation

One of the main design challenges related to the implementation of RODL was the need to support both live, dynamically changing research objects as well as immutable snapshots that are intended for a longterm preservation. With this in mind, the RODL has a modular structure that comprises the access components, the longterm components and the controller that manages the flow of data (see figure 9). For immutable research objects, they are stored in the longterm preservation repository once they are created. The live research objects, on the other hand, are pushed asynchronously after every change or periodically, depending on the configuration.

The access components are the storage backend - dLibra [**?**] - and the semantic metadata triplestore. dLibra provides file storage and retrieval functionalities, including file versioning and consistency checking. It has a built-in text search engine and it manages users and controls their access rights. It allows organizing stored objects into hierarchical structures and associating metadata at the level of object aggregations. It is also possible to use a built-in module for storing research objects directly in the filesystem.

The semantic metadata are additionally parsed and stored in the triplestore backed by Jena TDB [**?**]. Jena TDB is an actively developed RDF store implementation, which provides good support for transactions, querying, cacheing and using named graphs. The use of a triplestore helps in RODL internal data processing and offers a standard query mechanism for RODL clients. It also provides a flexible mechanism for storing metadata about any component of a research object that is identiable via a URI, which apart from workflows and other resources, may include parts of workflows or external resources (e.g. web services, data sources).

The UML sequence diagrams illustrate the interactions between the controller, the storage backend and the triplestore for the basic operations of creating a research object and aggregating resources to it (10,11,12,13,14). Creating immutable snapshots of research objects is a more complex process which involves copying the resources, recording their provenance, optional modifications by the user and finally releasing as a published, immutable object. Figure 15 shows how RODL clients can perform these steps via the RO Evolution API. Figures 16,17 present the interaction between internal RODL components when performing the process of creating the snapshot.

The longterm preservation component is built on dArceo [**?**] - a system for longterm preservation of digital objects developed by PSNC. dArceo stores the objects and monitors their quality, alerting the administrators if necessary 18. The standard monitoring activities include file format decay alerts and fixity checking but can be enhanced using a plugin mechanism. In case of RODL, dArceo periodically monitors the quality of research objects by calling the Checklist Evaluation and Stability Services [**?**, **?**] 19. If a change in quality is detected, notifications are generated as Atom feeds in compliance with the Notification API mentioned above. This helps detect and prevent workflow decay which occurs when an external resource or service used by the workflow becomes unavailable or is otherwise behaving differently.

dArceo gives the possibility to define migration plans that allow to perform a batch update of resources from one format to another, when necessary. In case of workflows, this may be applied for instance when a flat Taverna t2flow format should be converted to a complex scufl2 format (which, notabene, uses the RO model similarly to research objects). Other case could be a batch update of workflows that depend on a malfunctioning external resource.

Objects in dArceo can be stored on a range of backends, including specialized preservation repositories such as the Platon service [**?**], storing data in geographically distributed copies and guaranteeing their consistency.

A running instance of the RODL is available for testing at `http://sandbox.wf4ever-project.org/rodl/`. At the moment of writing, it holds more than 1300 research objects.

3.2.3 RODL clients

The use of a REST API as the primary interface of RODL shows the need for clients that can facilitate the interaction with RODL for the users. To this moment, the following clients support some or all of the RO APIs implemented by RODL.

The reference client of RODL is **the RO Portal**, developed alongside RODL to test new features and expose all available functionalities. It is a web application running at `http://sandbox.wf4ever-project.org/portal`. Its main features are research object exploration and visualization; it also allows to create user accounts in RODL and generate access tokens for other clients. The RO Portal uses all APIs of RODL.

The development version of **myExperiment** (see Section 3.3)uses RODL as a backend for storing packs. It

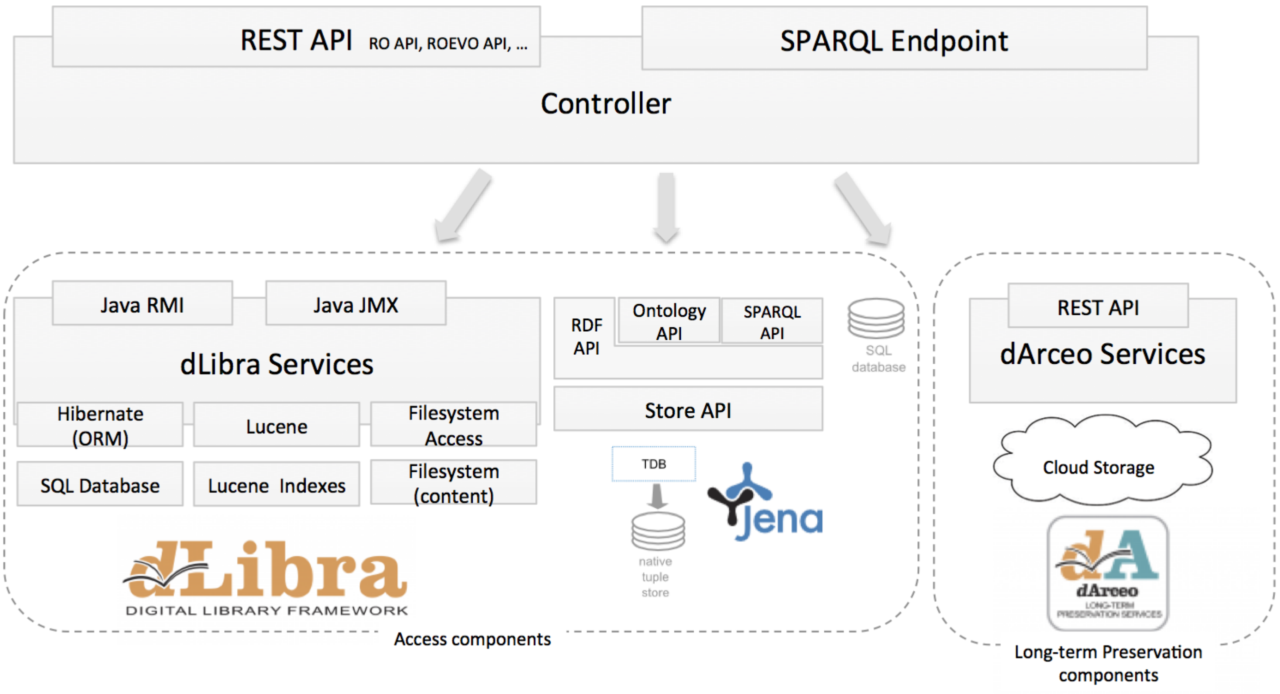


Figure 9: Research Objects Digital Library internal component diagram

uses the RO API. Finally, the **RO Manager** (see Section 3.1) is a command line tool that is primarily used to manage a research object stored on a local disk. It allows to push a research object to RODL via the RO

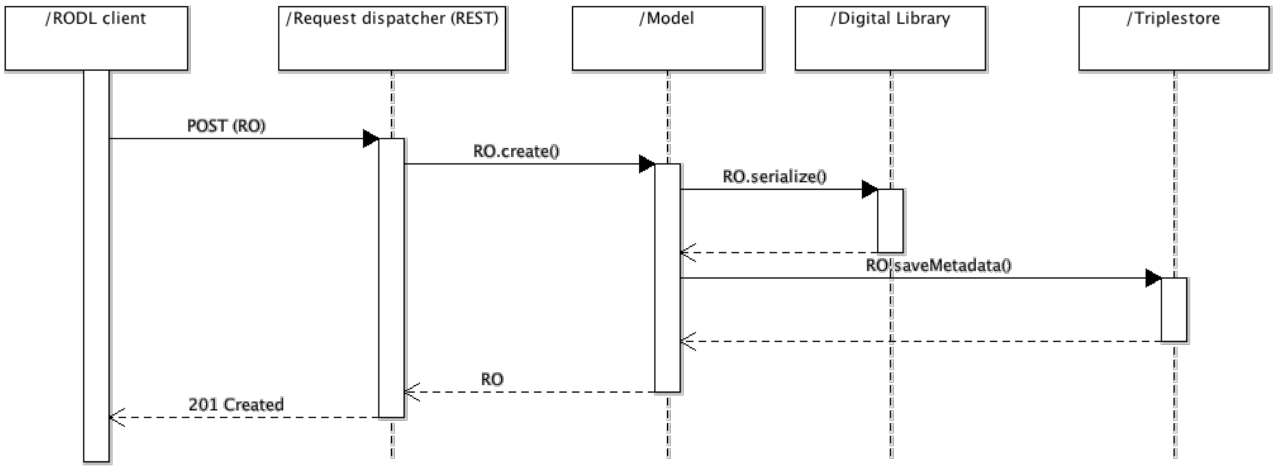


Figure 10: The sequence diagram for creating a research object in RODL

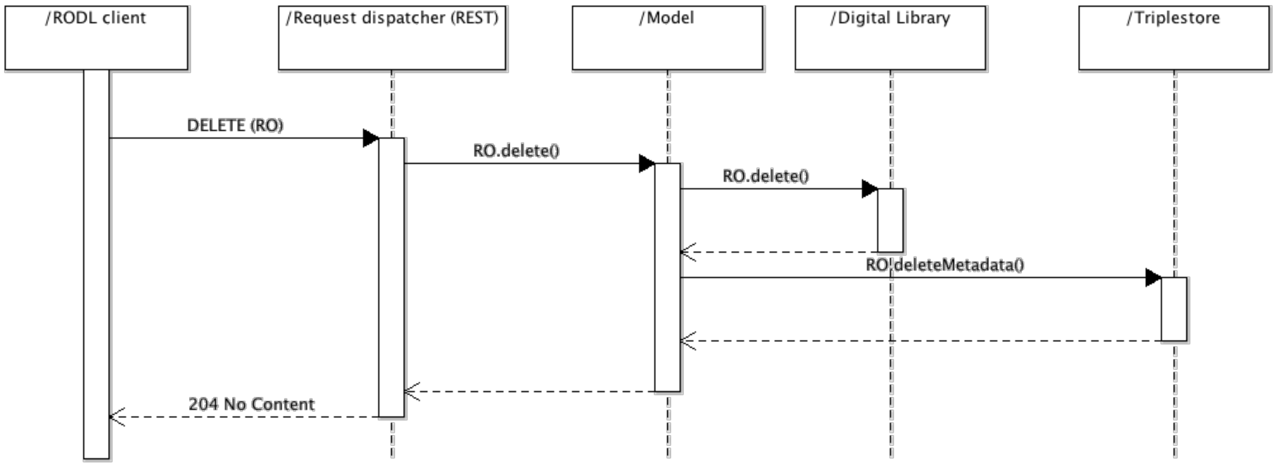


Figure 11: The sequence diagram for deleting a research object from RODL

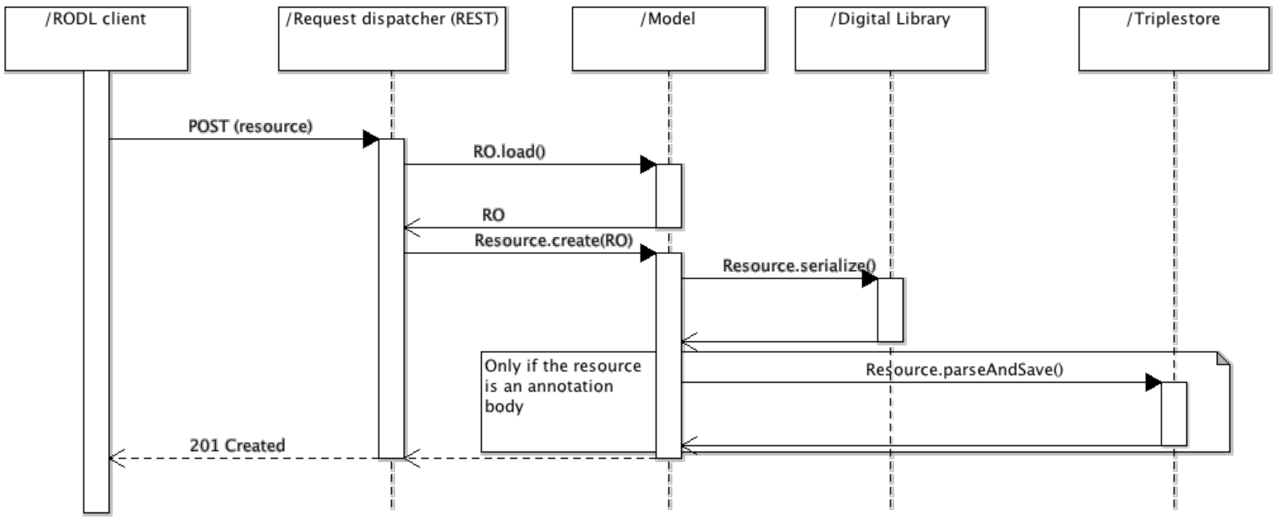


Figure 12: The sequence diagram for aggregating a new resource in a research object

API, as well as convert it into a snapshot in RODL.

3.3 Research Object-Enabled myExperiment

In this section, we describe how myExperiment [?] was extended in order to cater for the sharing, publication and curation of Research Objects. myExperiment is a virtual research environment targeted towards collaborations for sharing and publishing workflows (and experiments). It provides the functionalities necessary for sharing workflows within and across multiple communities. In doing so, myExperiment adopts a social web approach, which is adapted to the need of scientists. The workflows that are shared using myExperiment do not need to be specified in a particular workflow management system. For example, we find on myExpeirment workflows that have been specified using Galaxy [?], Taverna [?], Kepler [?] and Vistrails [?]. While initially targeted towards workflows, the creators of myExperiment were aware that scientists wants to share more than just workflows and experiments. Because of this, myExperiemnt was extended to support the sharing of artifacts known as Packs. A pack can be seen as a basic aggregation of resources, which can be workflows, but also files, presentations, papers, or links to external resources. The notion of packs have been widely adopted by scientists. At the time of writing, myExperiment had 337 packs. Just like a workflow,

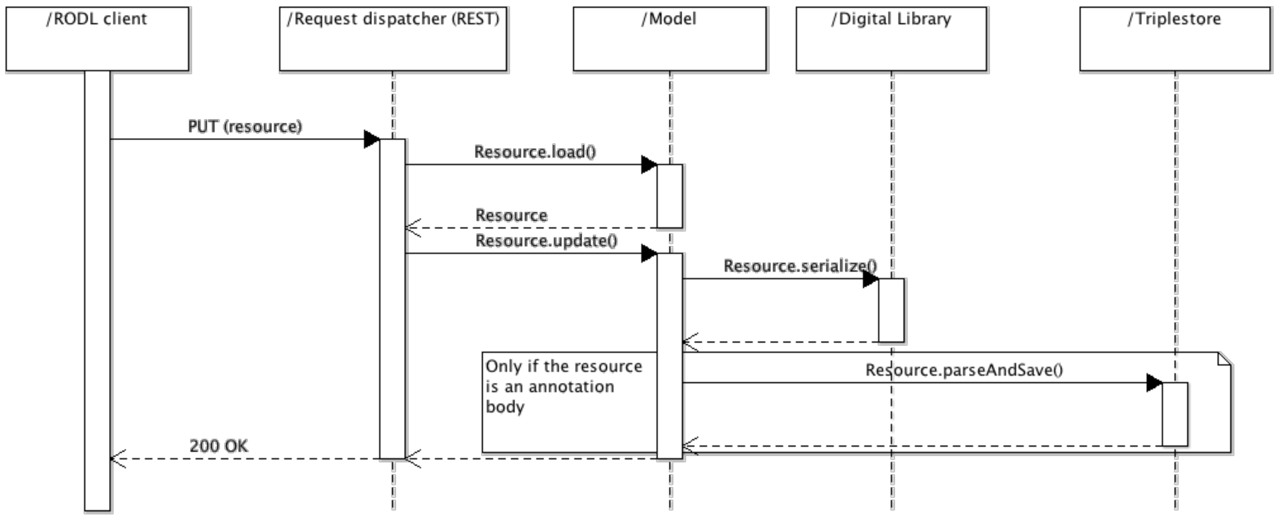


Figure 13: The sequence diagram for updating an existing resource in a research object

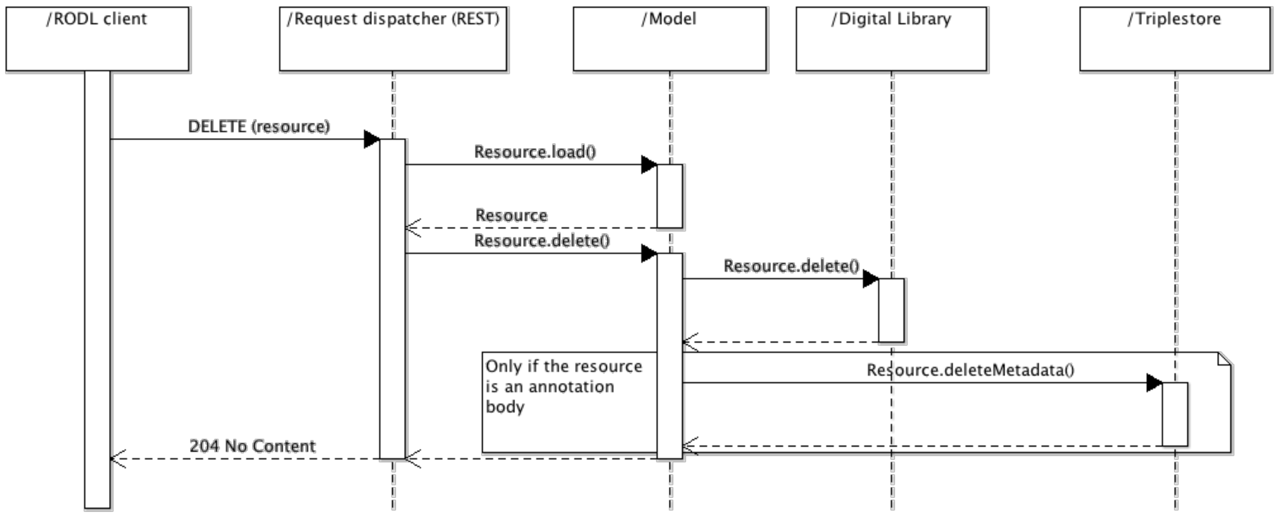


Figure 14: The sequence diagram for deleting a resource from a research object

using myExperiment a pack can be annotated and shared.

In order to support complex forms of sharing, reuse and preservation, we have worked during the last year on incorporating the notion of Research Objects into the development version of myExperiment [21]. In addition to the basic aggregation supported by packs, alpha myExperiment[22] provides the mechanisms for specifying metadata that describes the relationships between the resources within the aggregation. Moreover, the structure and the types of the resources that compose a pack are now inline with those that have been identifying thanks to the Research Object model. For example, a user is able to specify that a given file within a pack specifies the hypothesis, that another file specifies the workflow run obtained by enacting a given workflow, or that a given file states the conclusions drew by the scientists after analyzing the workflow run.

Figure 20 illustrates a high-level architecture of Alpha myExpeirment, the development version of myExperiment into which the Research Objects capabilities were incorporated. As illustrated in the figure, at the level of the Rail[23] model, data structures that represent the Research Object and associated resources have been incorporated. To manipulate such data structures, the controller layer has been extended, and to provide non information technology users with the ability to create and manage Research Objects, the view layer has been extended with the necessary HTML Web pages.

To illustrate how myExperiment can be used for managing Research Objects, Figure 21 depicts a sequence UML diagram illustrating a typical sequence of interactions that the user undergoes to create and share a Research Object. Alice (the user) first browses myExperiment to identify a workflow that is of interest to her investigation. Once she identified a relevant workflow, she downloads the wokflow, modifies and re-purposes it for her investigation. Once she is happy with the *new* workflow, Alice decides to create a Research Object. In doing so, she specifies the hypothesis within a file, which is stored within RODL. RODL acts as a back-end for myExperiment to store the information about Research Objects. Alice then upload her workflow to myExperiment. As a result, myExperiment sends a request to the `RO transformation service`, which uploads the workflow definition to RODL, transforms the workflow definition into wfdesc, and extracts the annotations that are bundled within the workflow definition. These elements, i.e., wfdesc specification and annotations, are then uploaded to the Research Object in RODL. Alice also uploads the workflow runs obtained as a result of enacting her workflow, and specifies the conclusion she comes to at the end of her investigation.

4 Workflow Abstraction using Motifs

Workflows serve a dual function: first, as detailed documentation of the scientific method used for an experiment (i. e. the input sources and processing steps taken for the derivation of a certain data item), and second, as re-usable, executable artifacts for data-intensive analysis. Scientific workflows are composed of a variety of data manipulation activities such as Data Movement, Data Transformation, Data Analysis and Data Visualization to serve the goals of the scientific study. The composition is done through the constructs made available by the workflow system used, and is largely shaped by the function undertaken by the workflow and the environment in which the system operates.

A major difficulty in understanding workflows is their complex nature. A workflow may contain several scientifically-significant analysis steps, combined with other Data Preparation or result delivery activities, and in different implementation styles depending on the environment and context in which the workflow is executed. This difficulty in understanding stands in the way of reusing workflows.

As a first step towards addressing this issue [**?**] describes a catalogue of domain independent conceptual abstractions for workflow steps called scientific Workflow Motifs. The catalogue was built based on an empirical analysis performed over 260 workflow descriptions from Taverna [**?**], Wings [**?**], Galaxy [**?**] and Vistrails [**?**]. Motifs are provided through i) a characterization of the kinds of data-oriented activities that are car-

[21] http://alpha.myexperiment.org/packs/

[22] It is worth noting that once the development in the myExperiment alpha is judged mature, the new functionalities will be staged to the production version of myExperiment.

[23] `http://rubyonrails.org`

ried out within workflows, which are referred to as Data-Operation motifs, and ii) a characterization of the different manners in which those activity motifs are realized/implemented within workflows, referred to as Workflow-Oriented motifs. Figure 22 shows an example of a Taverna workflow with its motifs highlighted.

This section describes the Workflow Motifs ontology[24], an OWL 2 encoding ot the aforementioned motif catalogue. The goal of this ontology is to provide the means to annotate workflows and their steps with the motifs of the vocabulary, without setting any restriction on how the workflows are defined themselves.

4.1 Representing Motifs

Figure 23 shows an overview of the class taxonomy of the ontology. The class `Motif` represents the different classes of motifs identified in the catalog. This class is categorized into two specialized sub-classes `DataOperationMotif` and `WorkflowMotif`, which are sub-classed following the taxonomy represented in [**?**].

The ontology provides three properties to link motifs to workflow specifications and their fragments. The `hasMotif` property associates workflows and their operations with their motifs. The properties `hasDataOperationMotif` and `hasWorkflowMotif` allow annotating workflows and their steps with more specificity. These properties have no domain specified, as different workflow models may use different vocabularies for describing workflows and their parts.

4.2 Representing Workflows and Workflow Steps

Workflows may be represented with different models and vocabularies like Wfdesc [**?**], OPMW [**?**], P-Plan [**?**] or D-PROV [**?**]. While providing an abstract and consistent representation of the workflow is not a prerequisite to the usage of the Motif ontology, we consider it a best-practice to use a model that is independent from any specific workflow language or technology. An example of annotation using the wfdesc model is given in Figure 24 by showing the annotations of part of the Taverna workflow shown in Figure 22.

The annotations encoded using the Motif Ontology could be used in a variety of applications. By providing explicit semantics on the data processing characteristic and the implementation characteristic of the operations, annotations improve understandability and interpretation. Moreover, they can be used to facilitate workflow discovery. For example, the user can issue a query to identify workflows that implement a specific flow of data manipulation and transformation (e.g., *return the workflows in which data reformatting is followed by data filtering and then data visualization*). Having information on characteristics of workflow operations allow for manipulation of workflows to generate summaries [**?**] of workflow descriptions or their execution traces.

5 Indexing Workflows

This section shows how workflows have been indexed using a generalized trie structure[25] and a serialization process. Workflows are a common way of providing and preserving scientific methods by explicitly encoding their processes. They are defined as directed acyclic graphs (DAG) and in Wf4Ever have been described by using both wfdesc and wfprov [**?**] vocabularies. Therefore, a workflow wf can be defined as a set of $Processes_{wf}$ and $Params_{wf} \mid wf = Processes_{wf} \cup Params_{wf}$ where the $Processes_{wf}$ are the definition of specific tasks to be executed, and $Params_{wf}$ defines the inputs and outputs of those tasks. On the other hand, a trie structure is an ordered tree data structure that allows to store dynamically a vector or an associative array where the keys are the values being stored itself. The main characteristic of this structure is absence of tree nodes being used for storing the key associated with that node but the position of the node within the tree defines the key. Another characteristic of this structure is that all the descendants

[24]http://purl.org/net/wf-motifs

[25]Trie comes from the word re**Trie**val indicating the process of information accessing but it is also called suffix tree.

of a node have a common prefix and therefore allows indexing simultaneously complete or partial paths to a specific node.

For our purposes of indexing a workflow, or generally speaking any DAG, by using a trie structure as a sequence of items which represents the DAG partially or completely for later accessing, a preprocessing of the set of workflows is needed as a first step. For accomplishing with this preprocessing goal of adapting a DAG to the trie indexing structure we have used one of the possible topological orders of a DAG. It is known that any DAG has at least one topological ordering which assures that if a vertex u is linked through an edge to vertex v, then after sorting it the vertex u will come before v in ordering. This topological ordering does not have to be unique and therefore a DAG could be defined by multiple topological sorts. Similarly to [?], in order to avoid the possibility of having similar workflows defined in different ways within the same indexing structure, we have chosen the lexicographical order of processes as common criteria to be applied. Therefore, the chosen overall used criteria for ordering the DAG sequentially has been both the topological and lexicographical orders.

The figure 25 shows how the "Extract proteins using a gi - output as fasta file" [26] example which was obtained from ProvBench data set [?], is sorted by applying our criteria. One of the main advantages of this approach is that after transforming the DAG workflow into a sequential linked set of resources it can be indexed almost instantaneously and the time for searching and exact matching is linear with the size of the tree which is also directly related to the vocabulary size of the domain.

5.1 Applications and Implementation

In order to provide scientists with searching and design capabilities we have implemented two different web services which makes use of the above introduced indexing structure for accessing to workflows and recommend next steps given the previous ones. It is worth highlighting that the implemented indexing structure could be also used for other purposes as the discovery of frequent patterns by using the collected statistical information and mining the semantic trie structure which would take full advantage of the intrinsic sequential ordering of the trie structure.
The implementation of both services (searching and next step recommendation) has been done applying the guidelines of the Wf4Ever project. Both services are REST and can be called by an HTTP GET method which upon on ACCEPT headers may return XML or JSON formats. Also for implementation purposes we have encapsulated the trie indexing structured in order to provided the above presented services.

5.1.1 Searching

This service `http://sandbox.wf4ever-project.org/wfabstraction/rest/search` searches for those workflows which contain a specific sequence of processes providing on real time a list including all of them. The service accepts an array of processes' names as input parameter process[] (e.g. process?=p1&p2&p3). Therefore the general call would be of the form: `http://sandbox.wf4ever-project.org/wfabstraction/rest/search{?process[]}`.

The output is an XML or JSON structure with the following attributes:

- **Process_id**: is the name of the process or processes used in the query.
- **freq**: is the number of times that the sequence of processes appears in other workflows.
- **URIs**: are the URIs of the workflows where the sequence appears.

[26] http://www.myexperiment.org/workflows/1182.html

5.1.2 Next step recommendation

The proposed trie structure captures the provenance of workflows execution associated to scientific experiments allowing their indexation based on the temporal information that they contain. The trie structure is also updated to gather the needed statistics for mining the execution of workflows and provide the recommended next process based on how frequent a pattern of use occurs. So far we collected the number of times a process occurs in the dataset and also the probability associated to that pattern given the set of patterns of the same size. The fact that the exact matching searching is linear with the size of the tree (which is dependent of the use vocabulary, in our case the domain has been restricted to the set of processes included in ProvBench [**?**]) makes it very suitable for this type of applications. The implemented service `http://sandbox.wf4ever-project.org/wfabstraction/rest/recommend` accepts an array of processes' names as input parameter process[] (e.g. process?=p1&p2&p3) and its general call would be of the form: `http://sandbox.wf4ever-project.org/wfabstraction/rest/recommend{?process[]}`

The output is an XML or JSON structure with different attributes:

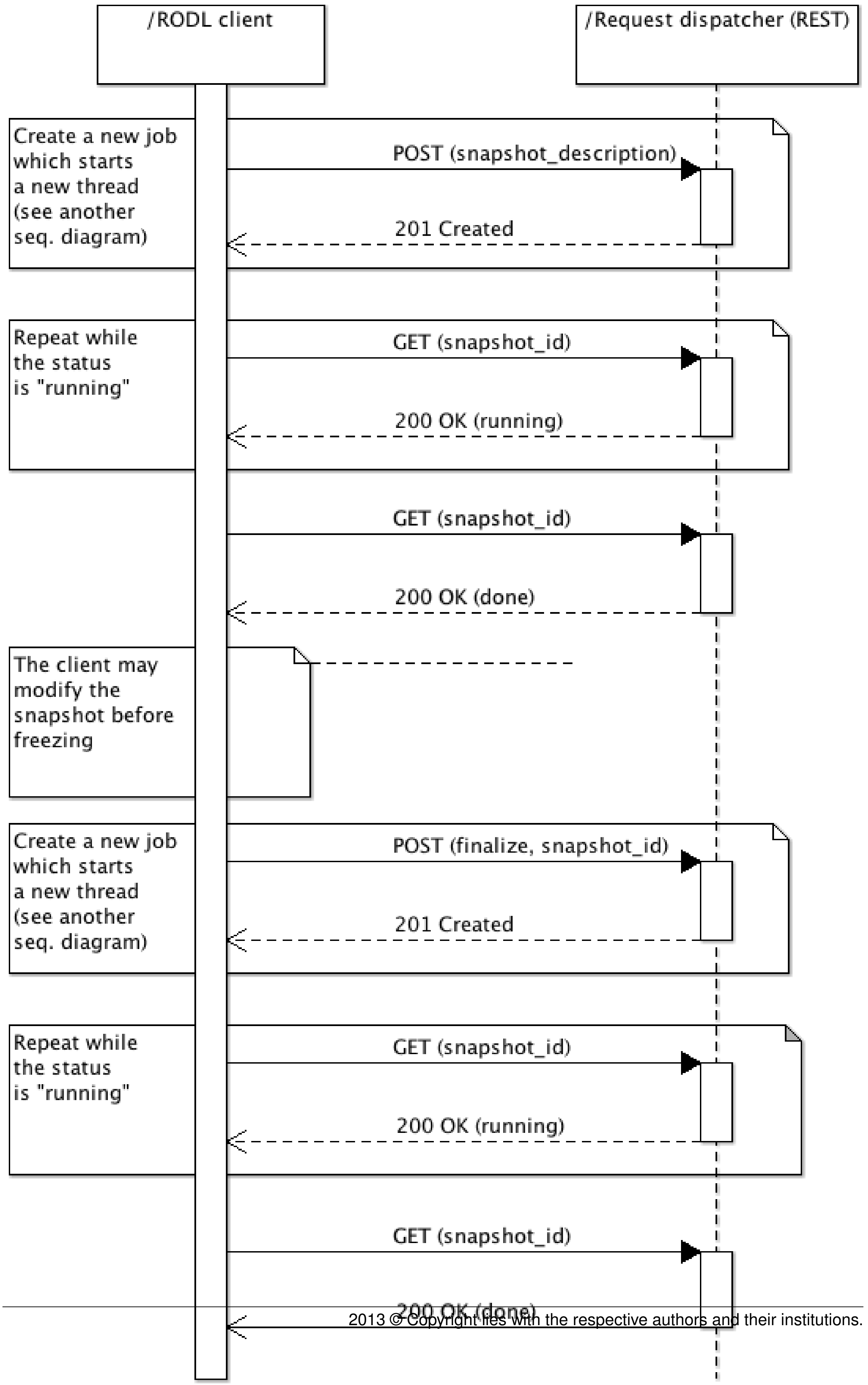
- **Id**: is the name of the recommended process for that input query.
- **freq**: is the number of times that it appears in different workflows.
- **prob**: is the probability of the given recommendation taking into account the whole set of possible next steps.

6 Summary

We have presented in this deliverable the final version Research Object model defined within Wf4Ever, as a family of ontologies. We also presented the tools that were built on the model in order to facilitate the creation, curation and sharing of Research Objects, namely, the Research Object Manager (RO Manager), a command line tool for creating, displaying and manipulating Research Objects, RODL, which acts as a back-end, with two storage alternatives: a digital repository to keep the content, as a triple store to manage the metadata content, and the myExperiment virtual research environment, which was extended to allow end-users to create, upload, share and curate Research Objects. We also presented two models that cater for advanced functionalities, namely abstracting and indexing workflows.

Our ongoing and future work aims to advertise and disseminate the Research Object model and the tools developed around it. In this respect, it is worth mentioning that we have launched a website dedicated to Research Objects[27], with examples that assist prospective adopters in understanding the model usage and benefits.

[27]`http://www.researchobject.org/`


/RODL client
/Request dispatcher (REST)
Create a new job which starts a new thread (see another seq. diagram)
POST (snapshot_description)
201 Created
Repeat while the status is "running"
GET (snapshot_id)
200 OK (running)
GET (snapshot_id)
200 OK (done)
The client may modify the snapshot before freezing
Create a new job which starts a new thread (see another seq. diagram)
POST (finalize, snapshot_id)
201 Created
Repeat while the status is "running"
GET (snapshot_id)
200 OK (running)
GET (snapshot_id)
200 OK (done)

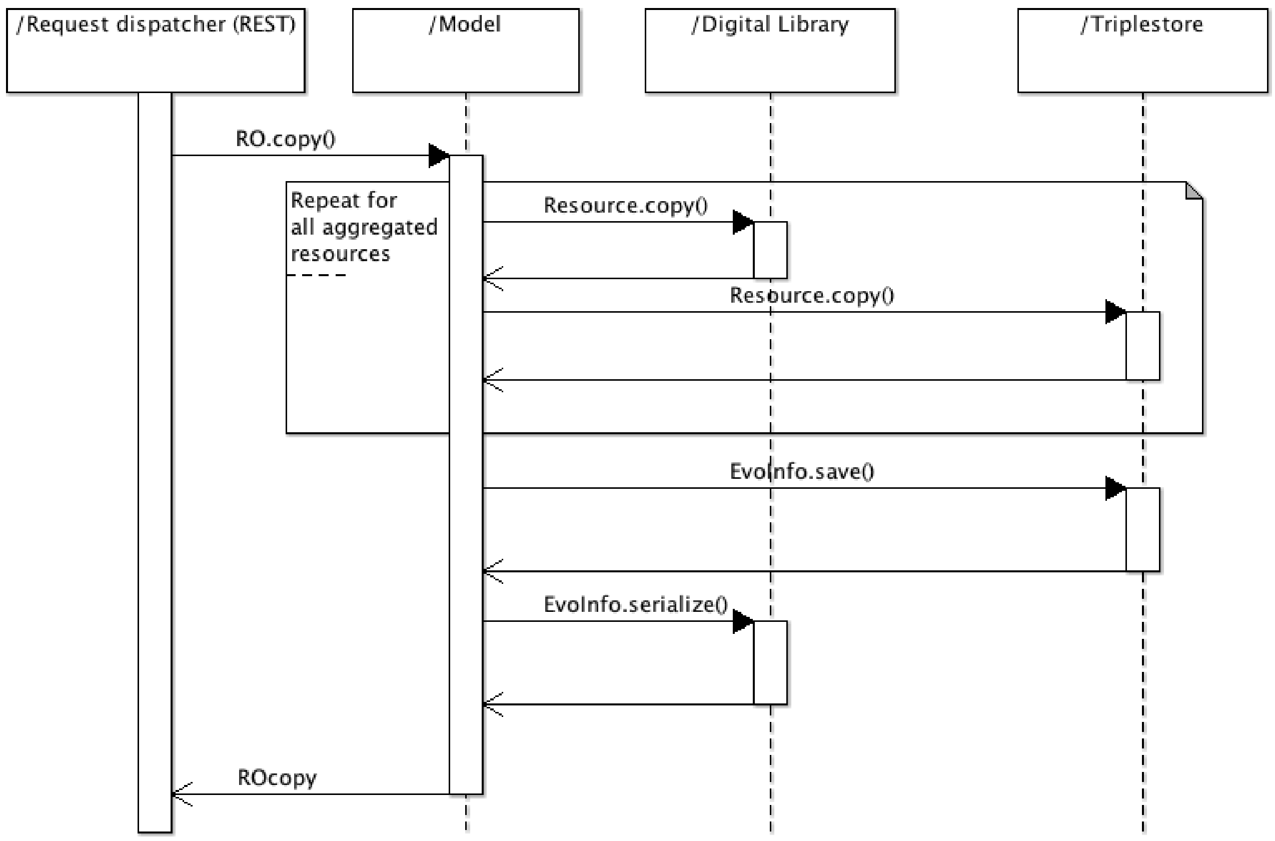


Figure 16: The sequence diagram for preparing the snapshot of a research object

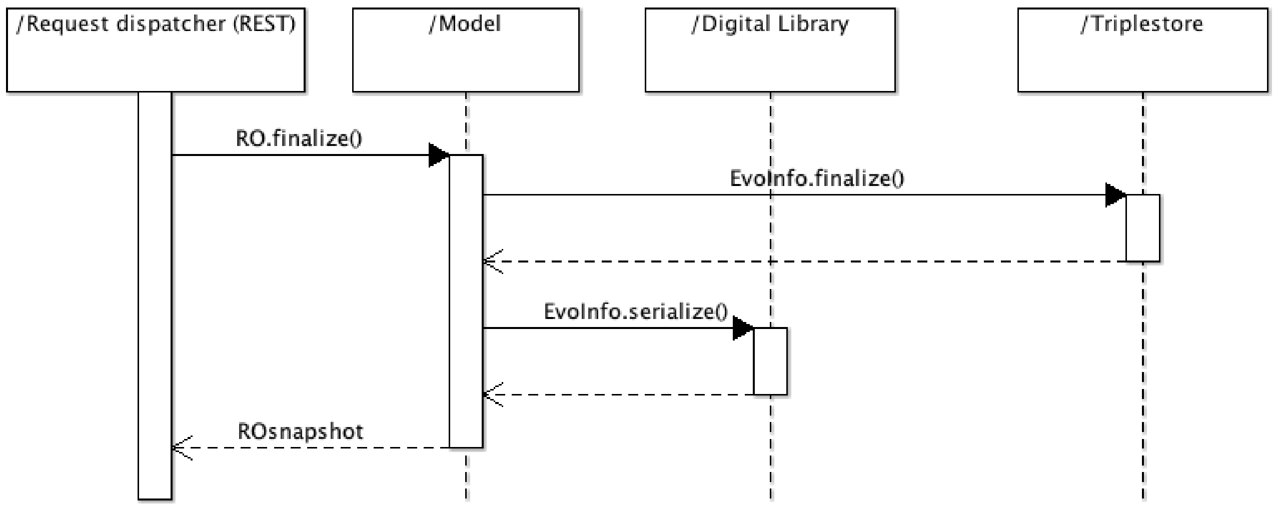


Figure 17: The sequence diagram for finalizing the snapshot and making it immutable

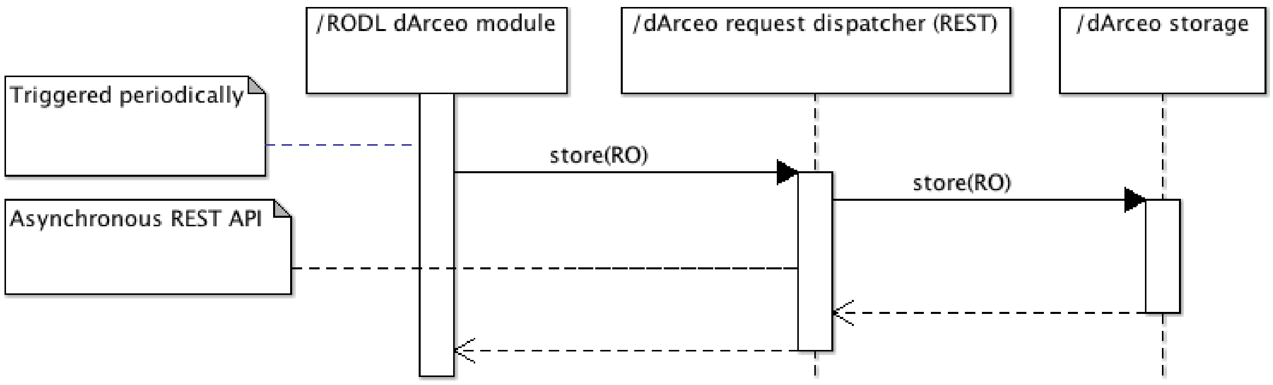


Figure 18: The sequence diagram for storing research objects in dArceo

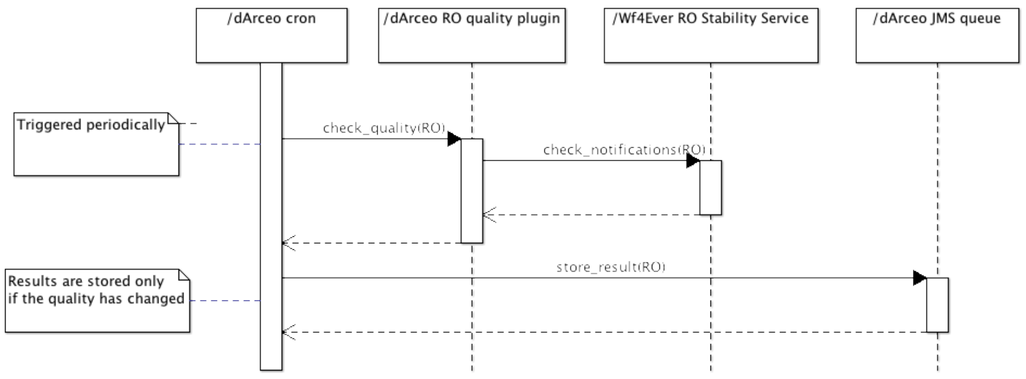


Figure 19: The sequence diagram for checking the quality of research objects in dArceo

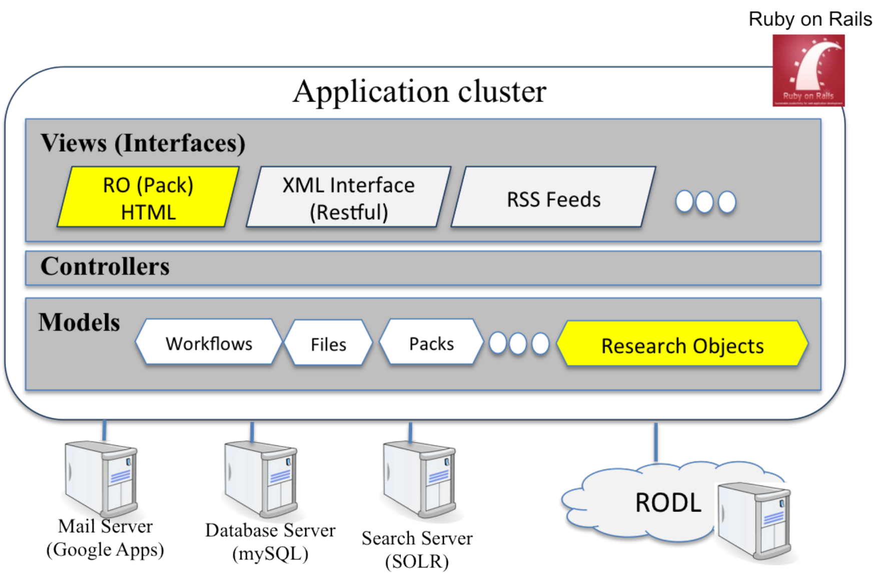


Figure 20: RO-enabled myExperiment.

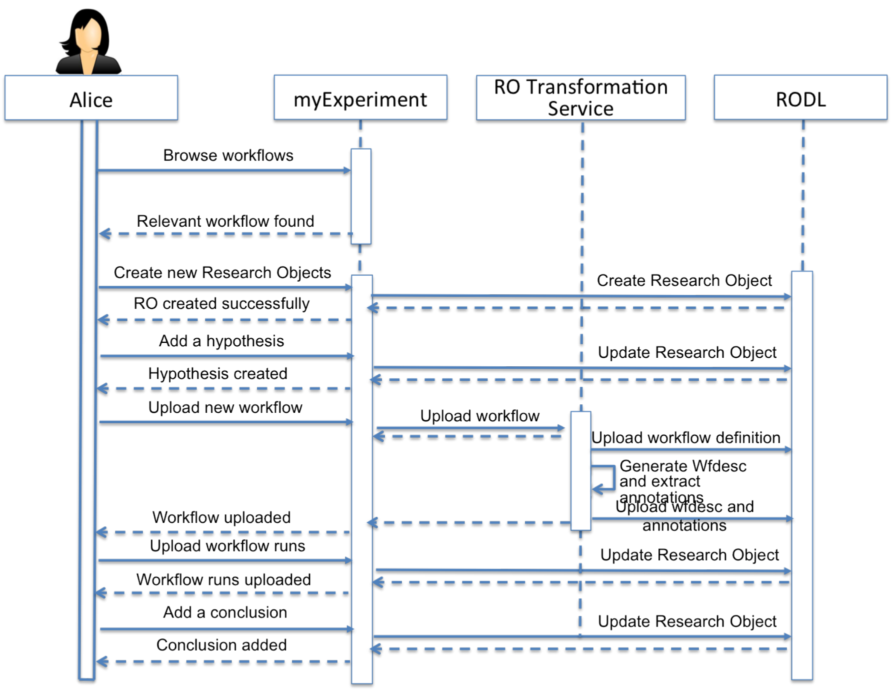


Figure 21: A Sequence diagram illustrating how myExperiment can be used to create Research Objects.

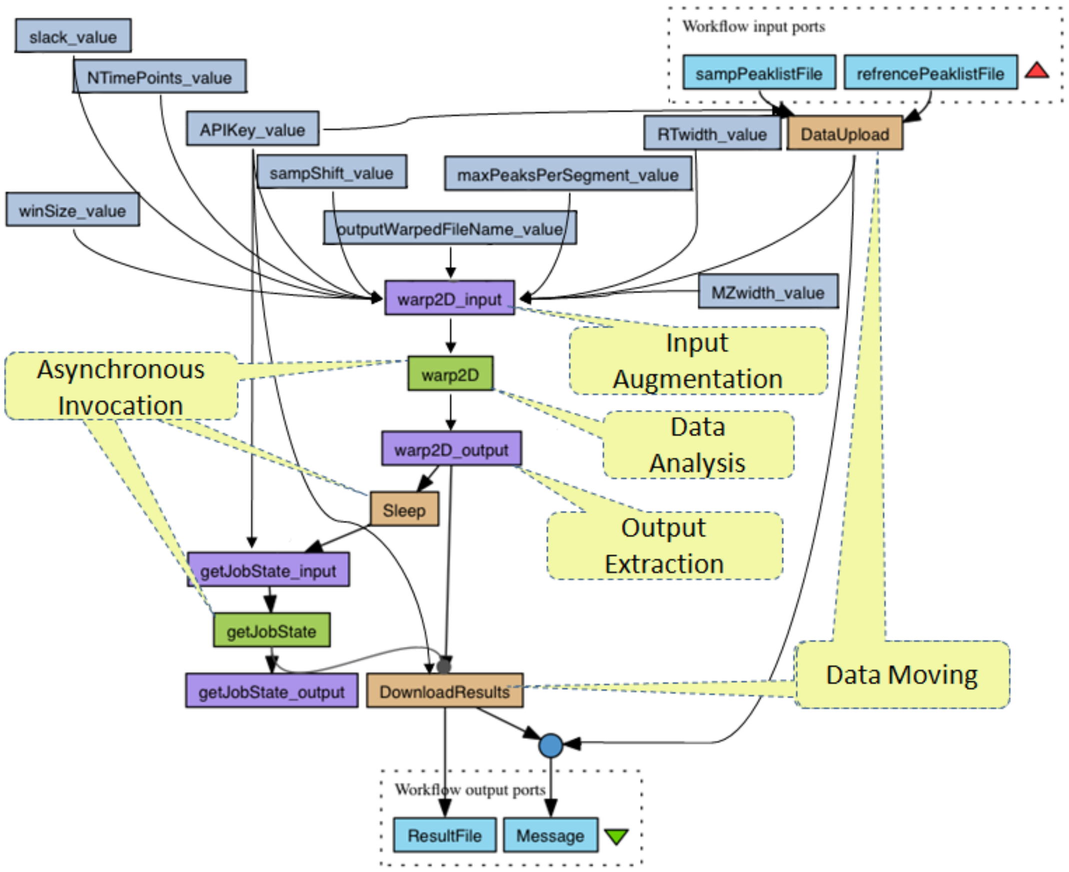


Figure 22: Sample motifs in a Taverna workflow for functional genomics. The workflow transfers data files containing proteomics data to a remote server and augments several parameters for the invocation request. Then the workflow waits for job completion and inquires about the state of the submitted warping job. Once the inquiry call is returned the results are downloaded from the remote server.

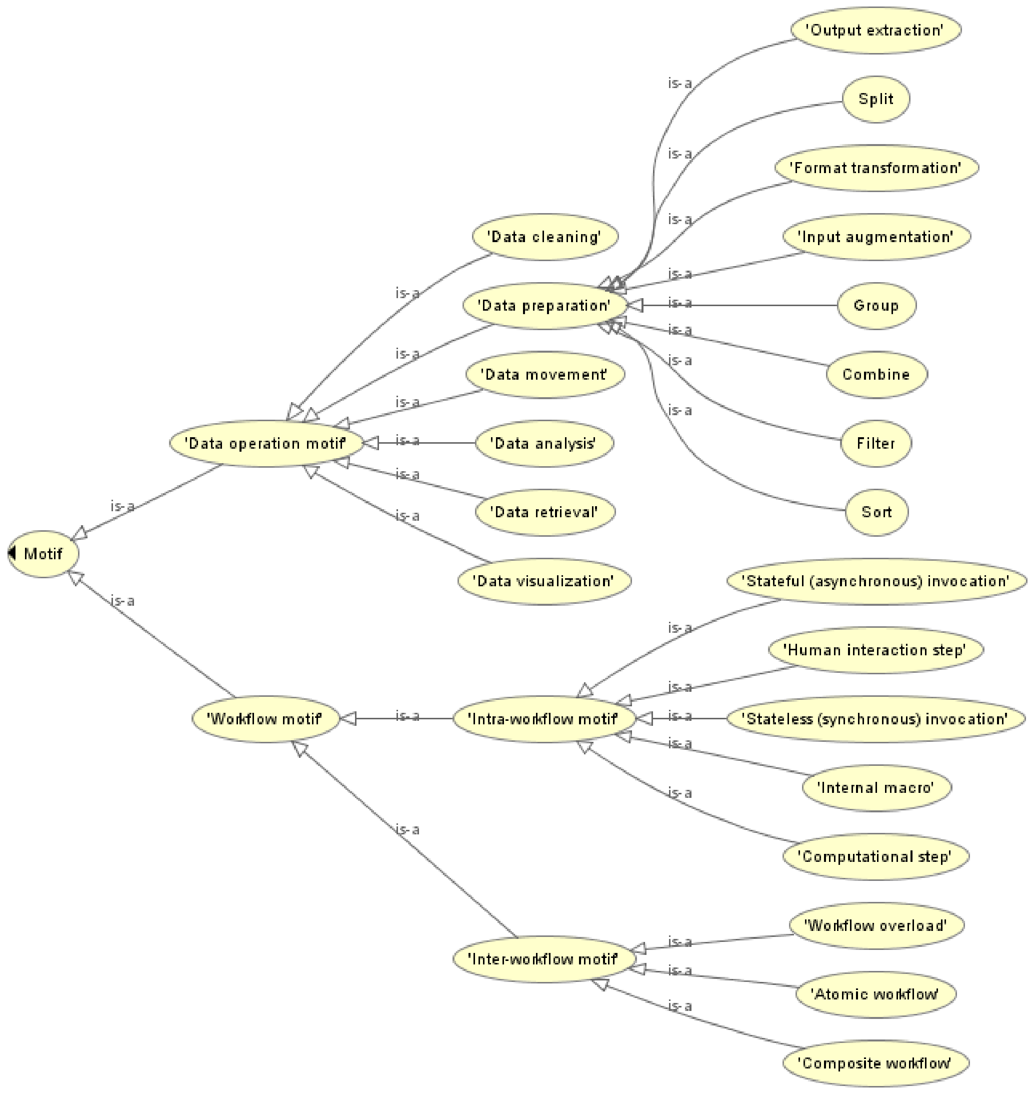


Figure 23: Diagram showing an overview of the class taxonomy of the motif OWL ontology.

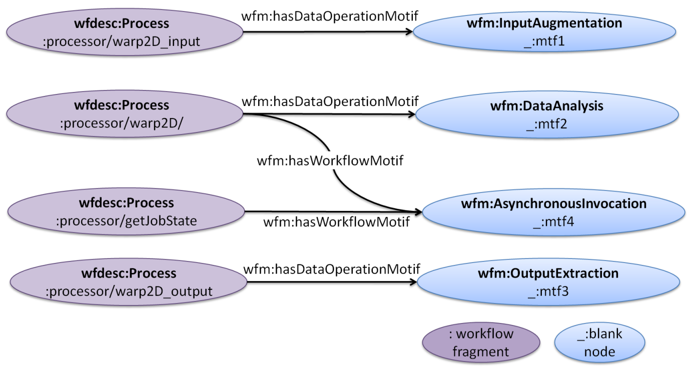
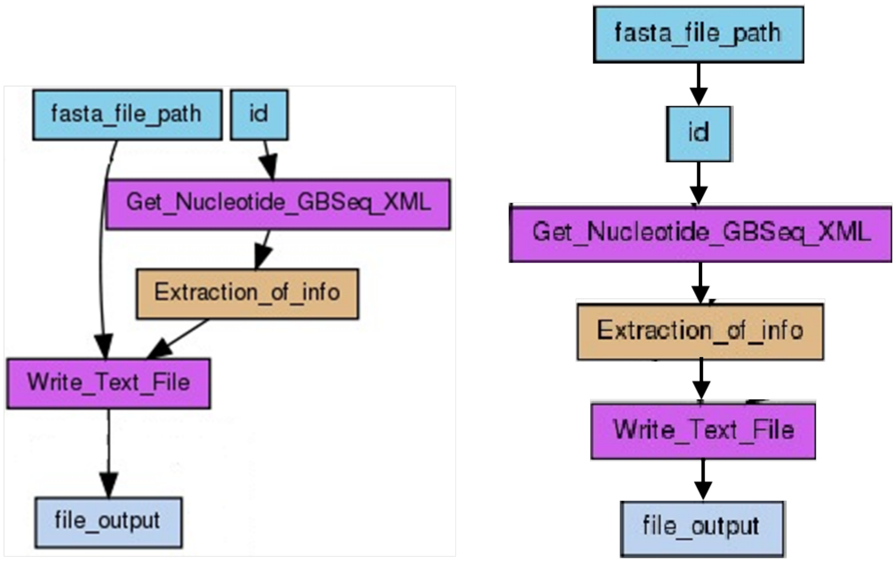


Figure 24: Subset of the annotations of the Taverna workflow shown in Figure 22 using the wfdesc model.



a) Workflow DAG b) Sequentially sorted workflow

Figure 25: Sorting the "Extract proteins using a gi - output as fasta file" example obtained from ProvBench

References