



**Wf4Ever: Advanced Workflow Preservation Technologies for Enhanced Science**

**STREP FP7-ICT-2007-6 270192**

**Objective ICT-2009.4.1 b) – "Advanced preservation scenarios"**

---

# D4.2: Design, implementation and deployment of Workflow Integrity and Authenticity Maintenance components – Phase II

---

**Deliverable Co-ordinator:** Esteban García-Cuesta

**Deliverable Co-ordinating Institution:** iSOCO

**Other Authors:** Graham Klyne (OXF), Aleix Garrido (iSOCO), Esteban García-Cuesta (iSOCO), Jose Manuel Gómez-Pérez (iSOCO), Jun Zhao (OXF).

This document describes the second phase of delivery of Integrity and Authenticity components implementation. It includes the description of the provenance models created, roevo which provides information about the RO evolution and wfprov which records the execution of workflows, and also describes the stability and completeness components for the evaluation of the quality of a RO.

| | | | |
|---|---|---|---|
| Document Identifier: | Wf4Ever/2013/D4.2v2/v1.0 | Date due: | 31/07/2013 |
| Class Deliverable: | Wf4Ever 270192 | Submission date: | 31/07/2013 |
| Project start date: | December 1, 2010 | Version: | v2.0 |
| Project duration: | 3 years | State: | Final |

| | | | |
|---|---|---|---|
| | | Distribution: | Public |

## 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:


| | |
|---|---|
| **Intelligent Software Components S.A.**<br>Edificio Testa<br>Avda. del Partenón 16-18, 1º, 7ª<br>Campo de las Naciones, 28042 Madrid<br>Spain<br>Contact person: Dr. Jose Manuel Gómez-Pérez<br>E-mail address: jmgomez@isoco.com | **University of Manchester**<br>Department of Computer Science,<br>University of Manchester, Oxford Road<br>Manchester, M13 9PL<br>United Kingdom<br>Contact person: Professor Carole Goble<br>E-mail address: carole.goble@manchester.ac.uk |
| **Universidad Politécnica de Madrid**<br>Departamento de Inteligencia Artificial<br>Facultad de Informática, UPM<br>28660 Boadilla del Monte, Madrid<br>Spain<br>Contact person: Dr. Oscar Corcho<br>E-mail address: ocorcho@fi.upm.es | **University of Oxford**<br>Department of Zoology<br>University of Oxford<br>South Parks Road, Oxford OX1 3PS<br>United Kingdom<br>Contact person: Dr. Jun Zhao / Professor David De Roure<br>E-mail address: {jun.zhao@zoo.ox.ac.uk, david.deroure@oerc.ox.ac.uk} |
| **Poznań Supercomputing and Networking Center**<br>Network Services Department<br>Poznań Supercomputing and Networking Center<br>Z. Noskowskiego 12/14, 61-704 Poznan<br>Poland<br>Contact person: Dr. Raúl Palma de León<br>E-mail address: rpalma@man.poznan.pl | **Instituto de Astrófísica de Andalucía**<br>Dpto. Astronomía Extragaláctica<br>Instituto Astrofísica Andalucía<br>Glorieta de la Astronomía s/n 18008 Granada,<br>Spain<br>Contact person: Dr. Lourdes Verdes-Montenegro<br>E-mail address: lourdes@iaa.es |
| **Leiden University Medical Centre**<br>Department of Human Genetics<br>Leiden University Medical Centre<br>Albinusdreef 2, 2333 ZA Leiden<br>The Netherlands<br>Contact person: Dr. Marco Roos<br>E-mail address: M.Roos1@uva.nl | |

## Change Log

| Version | Date | Amended by | Changes |
|---|---|---|---|
| 0 | 22-05-2012 | Esteban García-Cuesta | Outline included |
| 0.1 | 04-07-2012 | Esteban García-Cuesta | Initial draft included by adding information provided by the different authors |
| 0.2 | | | |
| 0.3 | | | |
| 0.4 | | | |
| 0.5 | | | |
| 0.6 | | | |
| 0.7 | | | |
| 0.8 | | | |
| 0.9 | | | |
| 1.10 | | | |
| 1.11 | | | |
| 1.12 | | | |

## Executive Summary

This deliverable includes the updated prototypes of the different integrity and authenticity (I&A) components of the project. This is the last of two deliverables regarding the design, implementation, and deployment of Workflow Integrity and Authenticity.

These components use the Research Object resources allocated in RODL (Research Object Digital Library) for evaluating the RO overall quality and providing some meaningful information for a better understanding of its current status. Among other useful information provided by these tools is worth to highlight the importance of collecting the provenance of the different quality dimension scores to provide a historical point of view of these measurements.

We also provide here a summary of the updated implementation of the I&A evaluation tool during phase II. In this second version of the deliverable we have focused on updating the two specific quality dimensions which were used to drive the design and implementation of our I&A evaluation tool, namely completeness and stability, and we also have contributed for the definition of a new dimension so called reliability which uses provenance of quality information for providing a more user oriented and meaningful information regarding the quality of an RO.

Regarding the provenance work, the models wfprov and wfdesc that we developed during the phase I did not need any further updates. Taking advantage of this fact, we have created a big corpus of provenance, so called BigProv, by using the wfprov ontology and the plugins implemented during the Phase I making it accessible to the community for benchmarking purposes.

Our updates on the current design and implementation of these three dimensions can be summarized in our new checklists designs, our new Minim model based on new specified requirements, an update on the evaluation software to use SPARQL1.1, accessing to RODL for retrieving the ROs which want to be evaluated, implementation of the new reliability dimension, and new visualization tools for these three dimensions. We furthermore started the evaluation process which will be also finished before M36 and fully included in the deliverable D4.3. “Final evaluation report of the workflow integrity and authenticity maintenance”. A glimpse of this evaluation process is also presented in this document by showing specific individual completeness’ dimension evaluations and also a simulated scenario based on real parameters for the validation of the reliability dimension.

## Table of contents

## List of Figures

## 1. Introduction

This document provides a precise description of the software components produced during phase II of Wf4Ever in the context of WP4 (workflow integrity and authenticity maintenance).

To be included:

- How the new implementations improve the previous quality end-user experience.
- How the new dimension reliability improves the previous quality end-user experience.
- How these new dimensions makes use of RODL and other wf4ever platform components for accessing to the ROs and needed functionalities (e.g. wfprov, OAI-ORE aggregates).
- We have focused on the main purposes of reuse and availability.

According to the DoW, this prototype will include the following functionalities: an updated Research Object provenance model that is the basis of the standardisation process in existing international initiatives (this was done for Y2), and extended methods for computing integrity and authenticity (we have improvement of stability and completeness and a new dimension reliability), taking into account different granularities (we use different minim models and different ways of evaluating reliability), and visualisation tools for them (we got new visualizations both for completeness, stability + reliability).

We have identified the following main contributions:

- Due to the advance state of the provenance vocabularies and the fact that they were early available to the project we have been able to create a PROV-Corpus based on Taverna and Wings workflow repositories so called BigProv. The main purpose of this corpus is to provide a suitable number of provenance of workflow results for benchmarking (e.g. extraction of macros, or identification of similar workflows based on their provenance of workflow results).
- Extending the current checklist-based approach for computing the completeness and the stability by: i) including a redefinition of the Minim model, ii) by adding a third dimension (reliability) based on the previous two mentioned, iii) by providing deeper granularity by extending the purpose criteria definition, and iv) storing and

providing accessibility to the provenance of the quality results as a resource of the RO.

The remainder of this document is structured as follows. Section 2 presents the provenance models and their alignment with the latest provenance standard from the World Wide Web Consortium, the PROV Ontology (PROV-O)[1]. Section **¡Error! No se encuentra el origen de la referencia.** describes our checklist-based approach, including the Minim data model, for representing the list of requirements that an RO must satisfy in order to be complete or stable. Section **¡Error! No se encuentra el origen de la referencia.** presents our current design and implementation of I&A evaluation components and how they can be integrated with components from other work packages in the context of the Wf4Ever architecture. Finally Section 6 presents our conclusions, providing a summary of this work and our plan for the next phase of the project.

**Technical Context**

During the implementation of the integrity and authenticity prototype we have made some decisions about the technical environment within which Wf4Ever is being deployed. These are:

- The system operates in the environment of the World Wide Web, supporting normal Web capabilities of retrieval, linking, etc. As such, URIs are used to denote arbitrary concepts, object types, etc. Concepts and entities manipulated by Wf4Ever are preferably identified using URIs
- Interfaces of the developed components have used HTTP/RESTful
- Research Objects (RO) are the main piece of information used which are the digitalization of a scientific experiment
- An RO contains metadata about provenance of its lifecycle, and also about its execution
- The provenance information has been modelled by the evolution ontology (roevo) and the provenance of workflow results ontology (wfprov) by using OWL[2]

[1] http://www.w3.org/TR/prov-o/

[2] http://www.w3.org/TR/owl-ref/

### Relation with Other WPs

Our work in WP4 about integrity and authenticity evaluation relies on different aspects that are treated elsewhere in the project. The main information units under study are ROs, whose representation is treated as part of WP2 work. Likewise, aspects about provenance dealing with RO evolution and versioning are addressed in combination with WP3. On the other hand, the evaluation of RO integrity and authenticity provides end users in WP5 and WP6 with valuable criteria to get some insight on the quality of ROs. There is also a strong relation with the overall integration of the project and user interfacing aspects like RO visualization, being addressed in WP1. Therefore, for a better understanding of the document we recommend it be read together with deliverables produced by other technical WPs, including D1.2v2 [1], D1.4v1 [2], D2.2v1 [3], and D3.2v1 [4].

# 2. Provenance

Provenance collects information about entities, activities, and people involved in producing a piece of data (in our project a research object), which among others can be used to form assessments about its quality, reliability or trustworthiness. An overview of a family of provenance information focusing in making them inter-operable can be found at PROV-Overview.

## 2.1. Provenance in Wf4Ever

In Wf4Ever there are two main types of provenance which have been modeled and used:

- **Provenance of workflow results**: providing a trace of the workflow processes, data resources and associated metadata that were used to produce the result of a workflow execution, and
- **RO Evolution**: as an underpinning for the representation of Research Object evolution (ROEVO), describing the evolution of resesearch objects over time, providing a record of the changes experienced in the different stages of their lifecycle.

The provenance of artifacts created by a workflow execution is captured during execution of a workflow by the workflow execution engine, and is published as annotations in a workflow RO. This provenance is expressed using the WFPROV ontology, which is part of the RO Model which also is in turn defined as a refinement of the W3C PROV-O ontology.

Regarding the provenance of the Research Object evolution, along with its possible origins in previous work, is captured through the Research Object Digital Library (RODL), and keeps track of the life cycle of an RO. This provenance is represented using the ROEVO ontology which also is defined as a refinement of the W3C PROV-O ontology.

We want to point out that the description of the WFPROV ontology and ROEVO ontology were introduced and described in D4.2v1 and can also be consulted there.

## 2.2. Provenance information in Research Objects

**Representing provenance in ROs**

To record provenance information in ROs we have used semantic annotations following the Annotation Ontology standard [Cicca'11]. That is, the RO includes RDF metadata

resources, containing provenance information, and these are identified as annotations of corresponding target resources by statements in the RO manifest. The Figure 1 shows the provenance of workflow results where the arrow labelled "RDF graph references" indicates that the provenance data contains direct references to the resource whose provenance is described. One such resource may describe provenance of multiple target resources, and an application that consults it does not need to know about the ro:annotatesAggregatedResource link in order to properly interpret the provenance information.

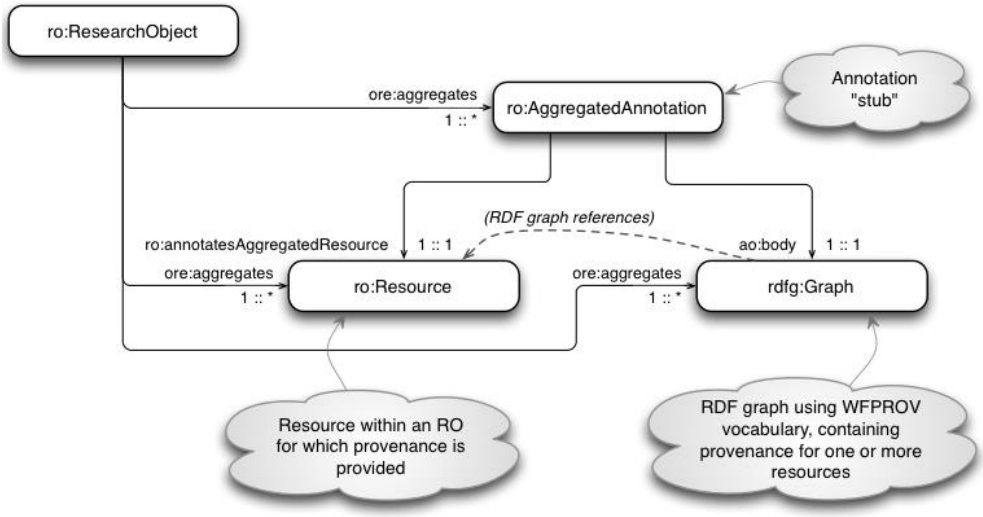



Figure 1 Provenance of workflow results

The provenance resource itself (the rdfg:Graph value) need not be part of the RO aggregation (i.e. it may be an external resource), but for practical purposes in our work an annotation body is generally treated as part of the RO aggregation.

The second type of provenance associated to Research Objects is captured in the description of Research Object Evolution (ROEVO). This type of provenance is expressed using a similar approach to that shown above, but with provenance relationships described between ROs, rather than between resources aggregated by an RO. Here, the ROEVO provenance resources capture the evolutionary relationships between a *Live* RO and its *Snapshots* or *Archives* states, and the forward looking relations are colour coded in blue, and the historical provenance relationships are colured in red as can bee seen in Figure 2. In the next we described how to access to this provenance.

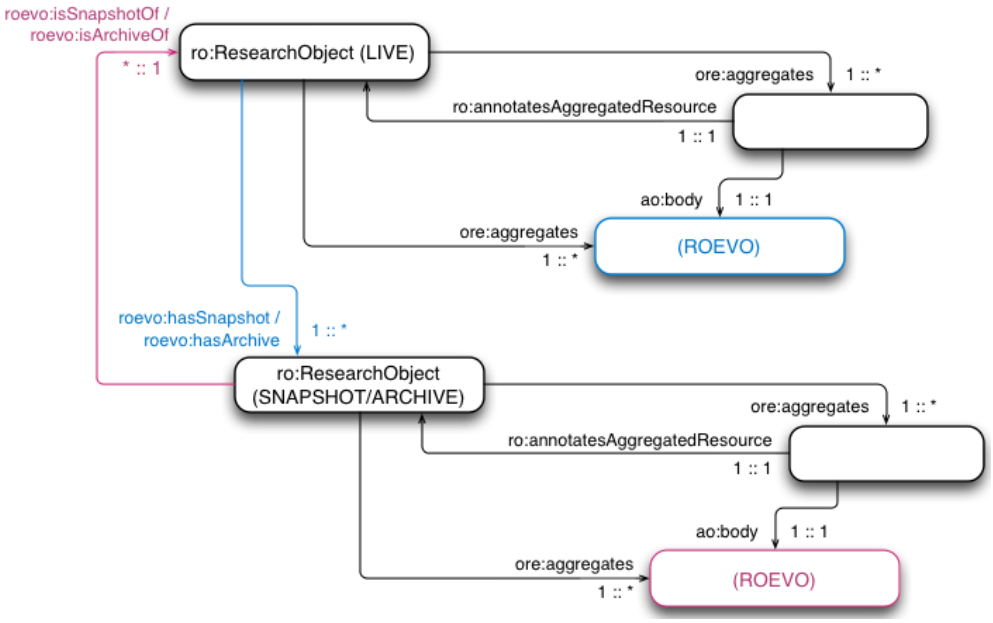



Figure 2 ROEVO provenance diagram

### Accesing provenance in ROs

Accessing provenance in an RO generally involves first reading the RO manifest, which contains information described in the diagrams above. Then, the RO manifest information is used to locate descriptions of the RO and its resources, which may include provenance and other information. The relevant information is read as one or several RDF graphs (annotations), from which the desired provenance information can be extracted.

For example, the checklist service reads all the annotations mentioned in the RO manifest, and creates a single RDF merged annotation graph of all the provenance and other information thus obtained. Provenance information can then be tested by suitably constructed SPARQL queries that are evaluated against the merged annotation graph.

Other applications may choose to be selective about the annotations they read, selecting those that are indicated in the RO manifest as having relevance to a particular target resource of interest.

So far it has been explained how to model provenance and how to access to that data once it is stored but we have not introduced how to obtain that data which has been mostly provided by Taverna[3].

### Taverna provenance export tools

Taverna executes workflows and therefore can capture provenance of workflow results, including individual processor iterations and their inputs and outputs. This provenance is kept in an internal database, which is used within the workbench to populate previous runs and intermediate results in the results view. The Figure 3 shows the current Taverna provenance architecture.

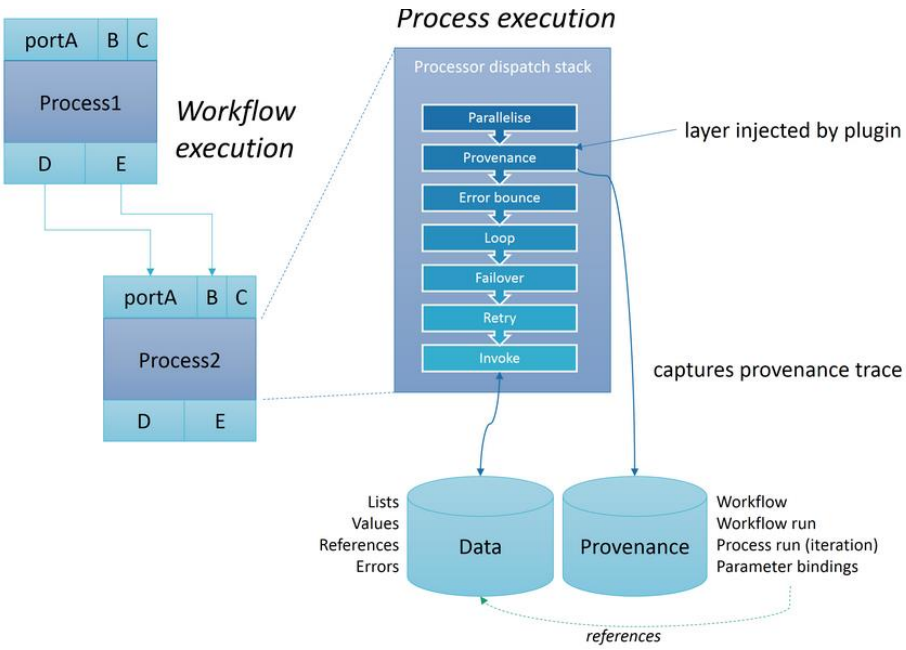



Figure 3 Taverna provenance architecture.

During execution of a Taverna workflow, the dispatch stack is responsible for the execution logic of an individual process invocaton, with layers like *parallelise* and *retry*. By injecting a provenance layer towards the top of the stack, a trace of each execution can be captured and stored in an internal provenance database. This includes a copy of the workflow definition, start/stop times for the workflow run and for each process execution. In addition

[3] http://www.taverna.org.uk/

the input and output parameters for every workflow and process execution is captured as references to Taverna's internal data store.

The provenance trace has been used by the implemented Taverna-PROV plugin to export the workflow run, including the output and intermediate values, and the provenance trace as a PROV-O RDF graph and a directory structure of the contents as individual files. The graph contents can be queried using SPARQL and processed with other PROV tools, such as the PROV Toolbox. The Taverna-PROV ontology extends the Wf4Ever wfprov ontology, which is based on PROV-O. Therefore no transformation (beyond OWL reasoning) is required within Wf4Ever to understand the created Taverna-PROV traces.

The Taverna provenance support was instrumental in making the Wf4Ever provenance corpus, a collection of 198 workflow run provenance traces from running 120 real world scientific workflows (more details can be consulted at A workflow PROV-corpus based on Taverna and Wings). This work was the motivation for forming the ProvBench initiative, launched at the BigProv workshop in 2013. Including the Wf4Ever provenance corpus, the first ProvBench accepted 8 provenance trace collections.

Example provenance traces, in addition to installation and usage instructions for the Taverna PROV export plugin are available at the taverna-prov project at GitHub.

To include the sequence diagram

## 2.3. Provenance applications

Within the Wf4Ever project, provenance information has been used for different purposes as it is described below:

**RO Portal**

The RO Portal displays RO evolution traces under the history tab of a Research Object page, based on stored ROEVO provenance information.

**myExperiment**

myExperiment will provide a high-level overview of WFPROV on each RO resource page. The overview will reveal if there are workflow runs in the research object and it will show at least the text based inputs and outputs for each run. The user should be provided with a

link to inspect the provenance trace with tools that are more suited to showing WFPROV metadata.

**Provenance visualization**

**Assessment of Keeg workflows**

Provenance information was used in the assessment of decay in KEGG workflows, specifically to locate the input data used to create additional RO annotations tested by the checklist evaluation.

For this purpose, provenance information was extracted from a Taverna-generated provenance trace using ASQC, a command-line SPARQL query tool. This can be seen the WF_Conversion.sh script used to convert KEGG workflows to ROs, in preparation to using the checklist service to perform decay detection.

**Workflow execution motifs**

(Motifs/Workflow abstraction - differencing using templates vs. provenance)

(Jun mentions paper in DropBox - FGCS-motif-extended-paper - )

(Also KCAP 2013)

(check no overlap with WP2 - Jun)

**Provenance summarization**

(Dropbox paper: bigdata 2013 - Small Is Beautiful: Summarizing Scientific Workflows Using Semantic Annotations)

(I note this work builds upon the motifs work - should they be covered together?)

(check no overlap with WP2 - Jun)

**Qualitity assessment**

The checklist evaluation service *may* query and test provenance values as part of a quality evaluation. In such cases, the provenance is queried like any RO annotation, with no special additional considerations.

## 2.4. Community engagement

**Provenance standardization in W3C**

The World Wide Web Consortium (W3C) effort to create a standard for provenance was started at about the same time as the Wf4Ever project, and completed its work in May of 2013. A full list of the working group documents produced is summarized in[PROV-Overview]. During this period, participants in the Wf4Ever project have been active participants in the working group, including as contributors to the key standards documents published:

- PROV-O - the PROV ontology, an OWL2 ontology allowing the mapping of the PROV data model to RDF PROV-O.
- PROV-DM - the PROV data model for provenance PROV-DM.
- PROV-N - a notation for provenance aimed at human consumption PROV-N.

We have also been co-editors of or contributors to a number of supporting working group documents, including PROV-PRIMER,PROV-AQ, PROV-DICTIONARY and PROV-DC.

These are not just committee-produced standards. At the time of their publication, there were over 60 documented implementations ([PROV-implementations]) of some aspects of PROV, most of which were producing or consuming elements of the provenance ontology (PROV-O), and some of which are already in deployed commercial products. The Wf4Ever project made significant contribution to this early adoption of the new provenance standards.

**ProvBench Challenge**

Several Wf4Ever project members were instrumental in setting up the ProvBench initiative, whose objective is to bootstrap the publication of provenance information in an open and

accessible fashion. The first ProvBench event was held at the 6th International Conference on Extending Database Technology (EDBT), as part of the First International Workshop on Managing and Querying Provenance Data at Scale (BIGProv'13). This inaugural event received 8 submissions from diverse interested research groups, including omne from Wf4Ever (see below).

**Wf4Ever provenance corpus**

BigProv collects 120 real provenance of workflows from the world scientific community. These workflows belongs to two different systems: Taverna and Wings and are associated to 12 different applications domains. The provenance traces have been specified by using the PROV-O ontology. The terms from other vocabularies as RO model and OPMW have been also used for the association between the provenance with their corresponding workflow description.

This dataset has been created for supporting the following scientific community interests and applications::

- Discovery of common "motifs" for annotation of workflows subgraphs by identifying the most frequent in-use patterns. This work can be consulted in the D2.2v2 "Design, implementation and deployment of workflow lifecycle management components– Phase II".
- Discovery of pattern similarities and linking similar scientific experiments.
- Identification of patterns of use for obtaining dependencies recognition.
- Verification of replicability of previously certified results.
- Include other uses....

This corpus has been generated by automatic capture of provenance from Taverna by using the provenance plug-in[4] which provides PROV-O output format. This plug-in was already implemented in its early stage at M20 and has been improved and tested for the generation of the BigProv corpus[5].

[4] http://wf4ever.github.com/taverna-prov/

4

[5] http://www.wf4ever-project.org/wiki/display/docs/Provenance+corpus

The Wf4Ever provenance corpus was assembled as a submission to the first ProvBench event.

The provenance traces were collected by executing 120 real-world scientific workflows. The workflows are from two different workflow systems: Taverna and Wings, covering 12 different application domains. Pfrovenance traces were provided for a total of 198 workflow runs, including 30 whose execution failed to complete for various reasons (unavailability of third party resources, illegal input values, *etc*.). Additionally, a number of provenance queries were assembled, to ask questions such as:

- What are the workflow runs available, and what is their start and end time?
- What are the workflow runs associated with a given workflow template, and how many of them failed?
- What are the workflow runs of a given workflow template, and what are the inputs they used and the outputs they generated?
- How many process runs are associated with a given workflow run, what is the start and end time of each one, and what are the inputs they used and the outputs they generated?
- Who executed a given workflow run?
- What are the services invoked as a result of a given workflow run?

Part of this corpus was used subsequently in our analysis of KEGG workflow decays, discussed in (@@xref section 4).

The provenance corpus data is available in Github, at https://github.com/wf4ever/provenance-corpus.

## 3. Quality Evaluation and Monitoring ROs in Wf4Ever

This section introduces the models designed and implemented in Wf4Ever which have provided the needed information for the establishment of a quantitative measure of the different dimensions identified as very important for the definition of overall quality RO criteria. Evaluating the health of the workflow contained in a specific research object requires transforming the additional information encapsulated by the research object into a quantifiable value and providing the scientists with the necessary means to interpret such values. We have established a clear separation between the different types of knowledge involved in order to evaluate the quality of a scientific workflow, as illustrated in Figure 4 which depicts a pyramid structured in three main layers, where the completeness, stability and reliability dimensions which helps to define the overall quality score of a research object is obtained through the evaluation of the information contained in the underlying levels.

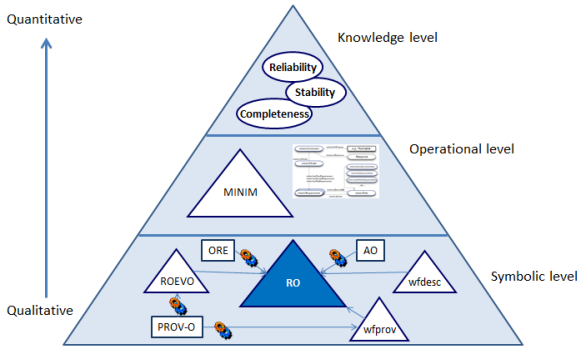



Figure 4 Quality ontology pyramid

The bottom layer spans across the main resources included in a research object and can be classified mainly as aggregations of several information resources, built on top of the ORE vocabulary and annotations which follows the Annotation Ontology. This layer corresponds to the RO model, described in the RO model specification RO model. This layer is also the placeholder of information related to the workflow included in the research object, in terms of the wfdesc ontology, and of the provenance of execution, following the wfprov ontology defined as an extension of the PROV-O standard. The ROEVO ontology is built upon wfprov the roevo ontology and enables the representation of the different stages of the RO life-cycle, their dependencies, changes and versions. Based on the metadata about the research object, its constituent parts and annotations, a new layer is included that contains knowledge about the minimum requirements that must be observed by the

research object in order to remain fit for a particular goal and about the predicates in charge of evaluating such requirements. This layer, which we call operational in the sense of the methods through which the requirements are evaluated, is modeled as checklists (see [zhao'12]) following the Minim OWL ontology. The evaluation of the checklists results into a number of boolean values indicating whether the specified requirements are fulfilled or not.

Finally, the top of the pyramid for assessing the reliability of scientific workflows contains quantitative values about reliability, stability, and completeness based on information derived from the outcomes of the checklist evaluation in the previous layer. These metrics are calculated following the algorithms and methods described in sections 4 and 5 and their values are stored as additional metadata in the research object, providing a compact type of quantitative information about the reliability of specific workflows. Based on these metrics plus the tooling necessary to interpret them scientists are enabled to make an informed decision about workflow reuse at the knowledge level, i.e. focusing on their domain expertise and not requiring a deep inspection of the information in the research object.

Regarding the main advances accomplished since M20 we want to highlight the implementation of the above introduced quality framework that unifies the two previously work on completeness and stability, and also includes the new dimension so called realiability. Also, the individual dimensions have been improved by incorporating new functionalities as it is explained in the next sections (e.g. new rules and tests), and a new set of presentations for visualizing the quality of a research object have been developed such as the new RO-Monitoring tool or the checklist verification service.

# 4. Completeness Evaluation

## 4.1. Introduction

Checklists are a widely used tool for controlling and managing quality assurance processes [CHECKLIST], and they have appeared in data quality assurance initiatives such as [MIBBI], which deals with coherent minimum reporting guidelines for scientific investigations. A checklist provides a measure of *fitness for purpose* rather than some overall measure of quality. We see this kind of fitness for purpose assessment as being of more practical use than a generic quality assessment, and indeed as the ultimate goal of any quality evaluation exercise. The suitability of a Research Object for different purposes may be evaluated using different checklists: there is no single set of criteria that meaningfully apply in all situations, which leads to a need to describe different quality requirements for different purposes. For this purpose, we have defined the Minim model [Minim-OWL].

Ideas for minimum information models developed for the [MIBBI] initiative have been adopted and generalized in our Minim model, which is an adaptation of the MIM model [MIM], to deal with a range of Research Object (RO) related quality concerns. Conforming to a minimum information model gives rise to a notion of *completeness*, i.e. that all information required for some purpose is present and available. In our work, a *checklist* is a set of requirements on a Research Object that can be used to determine whether or not all information required for some purpose is present, and also that the provided information meets some additional criteria.

The Minim model was introduced in [D4.2v1], reflecting its development as of August 2012, but its design and application have substantially progressed since then. In applying the checklist evaluation capability to myExperiment RO quality display, and other quality evaluations, we have:

- refactored the Minim model, and extended its range of capabilities to meed additional requirements,
- updated the checklist evaluation code to use a SPARQL 1.1 library in place of SPARQL 1.0l, significantly enhancing the expressive capability of the Minim model,
- developed a "traffic light" display of checklist results (for myExperiment integration and other uses),
- developed a REST web service for RO checklist evaluation, and deployed this in the Wf4Ever sandbox,

- created new checklist designs using the Minim model for myExperiment RO quality display, based on scenarios articulated by Wf4Ever project user partners, and incorporated checklist evaluation into work on RO stability and reliability evaluation (described below).

- We have also started work to evaluate the capabilities of the Minim model applied to a range of quality evaluation scenarios.

The sections that follow describe: the Minim data model used to define checklists, the Minim results data model used to express the result of a checklist evaluation, additional services created to support presentation of evaluation results to users of Research Objects, the checklist evaluation software structure and its integration with other Wf4Ever project elements, some applications that have been created using the checklist evaluation capabilities, and ongoing and future work relating to checklist-based quality evaluation.

## 4.2. Ontological models

A checklist to be evaluated is described by a Minim model, and the results of an assessment are presented using the Minim results model.

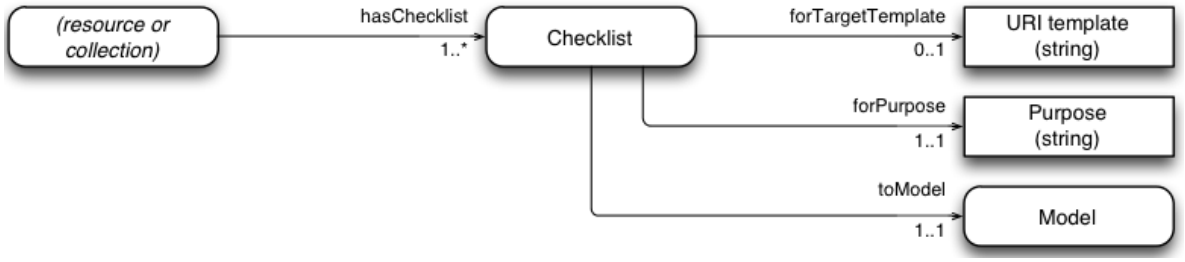
**Minim model for defining checklists**

This model has been significantly refactored and enhanced since that described in D4.2v1. The enhancements provide a cleaner structure to the overall model, greater expressive capability (including value cardinality tests similar to those supported my MIM), and clear identification of extension points at which new capabilities can be added to the model. The refactoring is done so that old-style Minim definitions do not conflict with new style definitions, and both may be supported in a single implementation.

The Minim ontology, its specification and its OWLDoc documentation are maintained in a GitHub project.

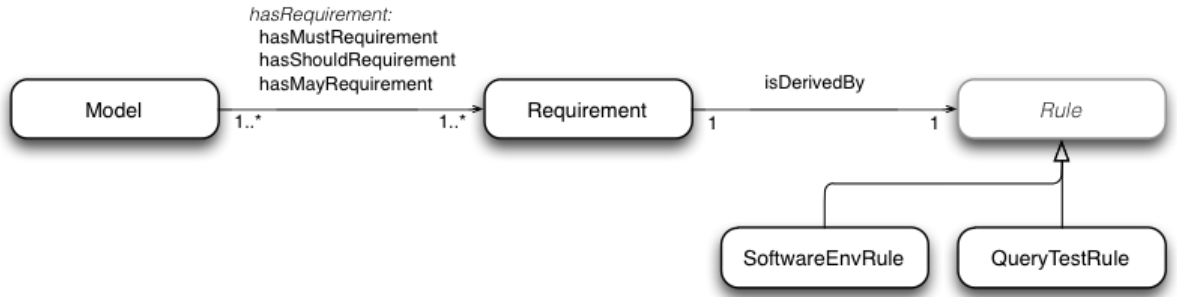
The main elements of the Minim model are:

- **Checklists** (or Constraints): Different models may be provided for different purposes; e.g. the requirements for reviewing an experiment may be different from those for a workflow to be runnable. A Minim `Checklist` associates a Minim `Model` with a description of the quality evaluation it is intended to serve.



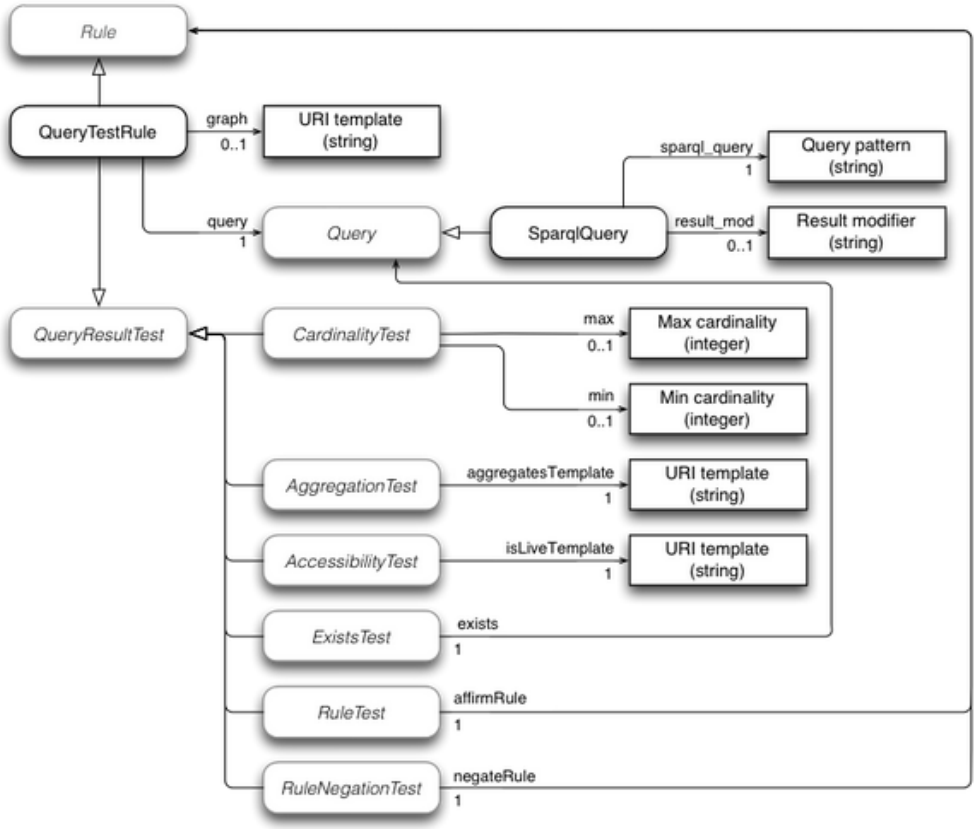

- **Models**: a Minim `Model` defines a list of `Requirement`s to be satisfied, which may be mandatory (`hasMustRequirement`), desirable (`hasShouldRequirement`), or optional (`hasMayRequirement`).

- **Requirements**: these denote some requirement to be satisfied by a Research Object, such as the presence of certain information about an experiment, or additional critreria to be satisfied by the available data. For example, we may wish to test not only that a suitable reference to input data is provided by an RO, but also that the data is live (accessible), or that its contents match a given value (integrity).




- **Rules**: a rule is associated with each requirement, and describes how the requirement is tested. A small number of different rule types are currently supported by the checklist service, including tests of the local computing environment for presence of particular software, and tests that query a Ressearch Object and perform tests on the results obtained. A rule determines whether a Research Object satisfies some technical requirement (e.g. that some specific resources are available, or accessible), which is interpreted as an indicator of some end-user goal.

- The main type of rule currently implemented is a `QueryTestRule`, which performs a query against the combined metadata (annotations) of an RO, and tests the result in various ways:




There are three classes defined by the Minim model that may be further subclassed to add new testing capabilities:

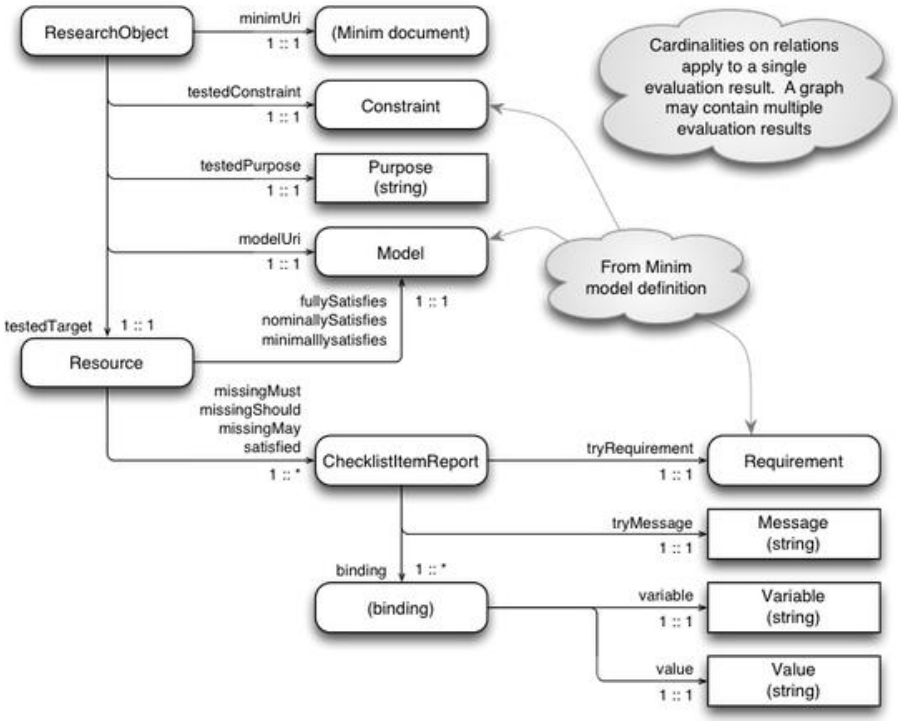
- `Rule`: new rule types can be introduced to perform tests for new kinds of requirement that cannot be handled within existing structures. For example, if a workflow has a dependency on a particular kind of computing hardware environment, such as a particular model of quantum computing coprocessor, then new rule types might be introduced to cover tests for such things.
- `Query`: this is an extension point within `QueryTestRule`, which allows query types other than SPARQL to be introduced. For example, a SPIN query processor, or an OWL expression used to find matching instances in the RO metadata might be introduced as different query types. The model assumes that query results are returned as lists of variable-binding sets (e.g. lists of dictionaries or hashes).
- `QueryResultTest`: this is another extension point within `QueryTestRule`, which allows different kinds of test to be applied to the result of a query against the RO metadata. For example,

checking that a particular URI in the metadata is the access point for an implementation of a specific web service might be added as a new query result test.

Any such extensions would need to be supported by new code added to a checklist evaluation implementation.

The outcome of a checklist evaluation is returned as an RDF graph, using terms defined by the Minim results model. The result graph returned also includes a copy of the Minim description used to define the assessment, so should contain all information to create a meaningful rendering of the result. The design is intended to allow multiple checklist results to be merged into a common RDF graph without losing information about which result applies to which combination of checklist, purpose and target resource.

The central values returned are the satisfaction properties (`missingMust`, etc.) that relate a target `Resource` with a Minim `Model` or the properties that relate a `Resource` to individual `ChecklistItemReport` (`satisfied`, `missingMust`, etc.). Additional values are included so the result graph contains sufficient information to generate a meaningful user presentation of the checklist evaluation result.




The main result of a checklist evaluation is an indication of whether a target resource `fullySatisfies`, `nominallySatisfies` or `` `minimallySatifies `` a checklist, evaluated in the context of a particular research object. `fullySatisfies` means that all MUST, SHOULD and MAY requirements are satisfied; `nominallySatisfies` means that all MUST and SHOULD requirements are satisfied, and `minimallySatifies` means that all MUSTrequirements are satisfied. Thus, the satisfaction properties forkm a hierarchy. If any MUST requirement is not satiusfied, then nine of the relations hold between the target resource and thge checlist.

A breakdown of the checklist evaluation result is given by a number of `missingMust`, `missingShould`, `missing Masy` and/or `satisfied` properties, which indicate the evaluation result for each individual checklist item as a relationship between the target resource and the corresponding checklist requirement. Associated with each of these individual item results is a message string which gives an explanation of the outcome of the test for that item. Also associated with the result for each checklist item there may be one or more variable bindings which may provide more detailed information about the reason for success or failure of the test. Typically, the values are associated with the result of a query performed against the Research Object. For example, a test for liveness of workflow inputs may use a query to find the workflows and

associated inputs, and a test for liveness of an input is performed for each result of that query; when one of those inputs is found to be not accessible, the corresponding query result variables are returned as part of the checklist item result.

This is an example Minim requirement that tests for presence of a synonym in chembox data:

```
:Synonym a minim:Requirement ;
  minim:isDerivedBy
    [ a minim:QueryTestRule ;
      minim:query
        [ a minim:SparqlQuery ;
          minim:sparql_query "?targetres chembox:OtherNames ?value" ;
        ] ;
      minim:min 1 ;
      minim:showpass "Synonym is present" ;
      minim:showfail "No synomym is present" ;
    ] .
```

This returns the following (partial) result for a target resource `http://purl.org/net/chembox/N-Methylformamide` for which no synonym exists:

```
<http://purl.org/net/chembox/N-Methylformamide>
    minim:minimallySatisfies :minim_model ;
    minim:nominallySatisfies :minim_model ;
    minim:missingMay
        [ minim:tryMessage "No synomym is present" ;
            minim:tryRequirement :Synonym ;
            result:binding
                [ result:variable "targetres" ;
                  result:value "http://purl.org/net/chembox/N-Methylformamide" ],
                [ result:variable "query" ;
                  result:value "?targetres chembox:OtherNames ?value" ],
                [ result:variable "min" ;       result:value 1 ],
                [ result:variable "_count";     result:value 0  ]
        ] .
```

The variable bindings here provide information about the failed checklist item test, including the target resource, the failed query, the number of query matches, etc.

## 4.3. Implementation and integration

The checklist evaluation service is implemented as part of the codebase for RO Manager [D2.2v2], which is implemented in Python, and is available as an installable package through the Python Package Index (PyPI) at https://pypi.python.org/pypi/ro-manager.

The source code is maintained in the Github project at https://github.com/wf4ever/ro-manager.

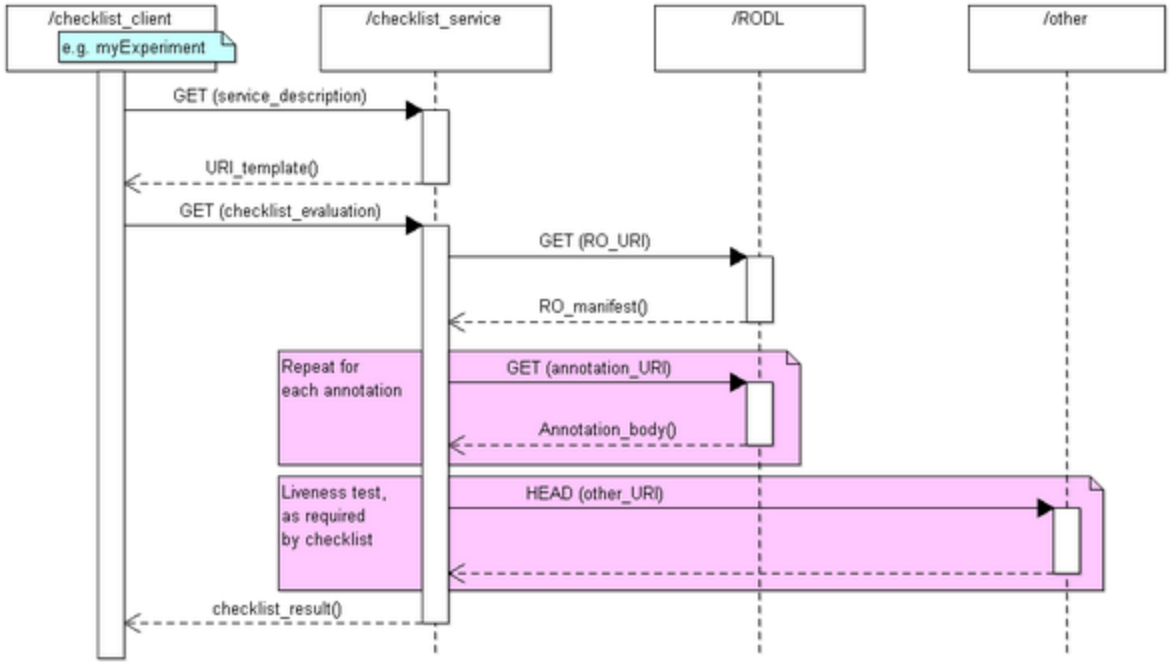
The checklist evaluation can be invoked by the RO Manager command line tool (`ro evaluate checklist` command), or by a separate web service that incorporates an in-built HTTP server based on the The Pyramid Web Application Development Framework, which is part of the Pylons project. The command line version of checklist evaluation is used mainly for development purposes; the discussions that follow will consider just the web service deployment.

**Minim model for defining checklists**

The Wf4Ever architecture is designed around use of linked data and REST web services, with interaction between components being handled by HTTP requests.

A checklist evaluation is invoked by a simple HTTP GET operation, in which the RO, Minim resource URI, target resource URI and purpose are encoded in the request URI. The evaluation result is the result of the GET operation. The Wf4Ever projhect wiki has a more complete description of the checklist API.

The checklist service in turn interacts with the RO in RODL (or in some other service offering the RO API elements used to access an RO), mainly to retrieve the RO annotations. Some checklist items, such as those that check for liveness of workflow dependencies, may cause further requests to arbitrary web resources named in the RO metadata.




### Source code organization

A fuller description of the source code organization is covered in [D2.2v2]. The key parts that are specific to checklist evaluation service are:

- `src/` - umbrella directory for the source code for RO Manager and the checklist evaluation service. Also contains the script used to create and share an installable package for the RO Manager tool.
- `iaeval/` - checklist evaluation functions, used by command line tool and evaluation web service.
- `roweb/` - the checklist evaluation web service, and traffic light display: this is mainly a web front end to functionality implemented by modules in `rocommand` and `roweb`.

### Key modules

Key modules that drive the execution of RO checklist evaluation web service are:

- `src/iaeval/ro_eval_minim.py` - checklist evaluation function (cf. `evaluate`)
- `src/iaeval/ro_minim.py` - Minim checklist definition access and parsing.

- `src/roweb/rowebservices.py` - web interface for checklist evaluation and "traffic-light" display functions.
- `src/roweb/RdfReport.py` - a simple RDF report generator that is used to generate HTML and JSON renderings of checklist results for "traffic light" displays
- `src/roweb/TraffiucLightReports.py` - contains report definitions, used in conjunction with `RdfReport.py`, to generate HTML and JSON renderings of checklist results for "traffic light" displays

**Dependencies**

The RO checklist evaluation is very heavily dependent on RDFLib, which provides RDF parsing, formatting and SPARQL Query capabilities. It uses the Pyramid web framework, and uritemplate for RFC 6570 template expansion.

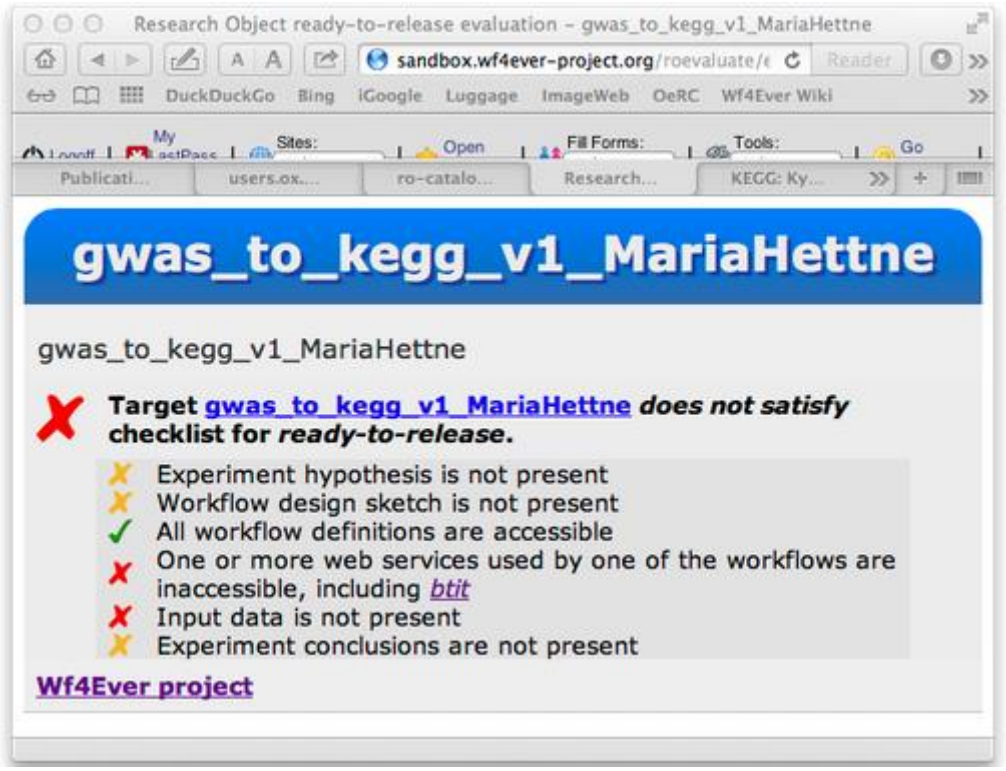
## 4.4. Applications

Within the Wf4Ever project, we have employed checklists in a number of areas.

**Detection of workflow decay**

A key goal for our work has been to anticipate and detect the potential causes of workflow decay.In 2012, the Kyoto Encyclopedia of Genes and Genomes (KEGG, http://www.genome.jp/kegg/) announced that they were introducing a REST interface for their discovery service, and discontinuing the older web Services based interface. There are a number of workflows in myExperiment that use the older KEGG services. This presented us with a golden opportunity to test our decay detection capabilities. Before the old service was shut down, the KEGG-using workflows were surveyed and a considerable number were found to still be executable, and a provenance corpus was created for each of these (https://github.com/wf4ever/provenance-corpus/tree/master/Taverna_kegg_wf).

Our hypothesis was that after the KEGG web services were shut down at the end of 2012, our checklist service should successfully detect and report the workflow decay. For example, the previous screenshot example now appears thus:




where *btit* is one of the KEGG web services that has been withdrawn.

This year, following closure of the KEGG web services, we converted as many as we could of the original workflows to Research Objects. About half could not be converted because they were based on an older form of Taverna workflow definition not supported by out workflow-to-RO tooling, but we were still able to convert more than 20 workflows to ROs - a useful number. We applied the checklist evaluation service to the resulting ROs. On our first attempt, we were able to successfully detect decay of 19 out of 21 workflows. We have since been able to fix the workflow descriptions of the remaining 2 so that their decay is also correctly detected.

The code, data and notes for this work is available from Github, at Kegg-workflow-evaluation.

This work is included in our submission to the ISWC 2013 in-use paper track.

**Completeness assessment for workflow decay prevention**

In earlier work [WF-decay], we analyzed causes of workflow decay for a sample of workflows selected from myExperiment. Based in part on the results of this investigation, and on consultation with our user partners about what additional information is helpful for facilitating re-use of workflows, we created a checklist that can be used to test the presence of information to support workflow re-use and repair, with a view that such a checklist can be incorporated into then practices of workflow creation and use to encourage experimenters to provide all useful information, and to automate some mechanical aspects of the review process that might otherwise have to be done manually.

This work led to checklist definitions such as checklist-runnable.rdf and workflow-experiment-checklist.rdf, which provide assessments similar to that shown in the screenshot above.

**Basis for stability assessment**

The work described above has considered static analysis of Research Objects, but it has also been used as the basis for work on dynamic analysis of Research Object stability and reliability, described later in this document.

**Completeness assessment of resource descriptions: chembox**

We have used checklists to evaluyate the completeness of resource descriptions extracted from external sources.Specicially, we used the checklist evaluation service to assess the completeness of chemical descriptrions in DBPedia, which in turn were extracted from Wikipedia "Chembox" templates.

The checklist used (chembox-minim-samples.ttl) and the script (chembox_evaluate.sh) used to perform the evaluations are available in Github. The chemical data extracted is in a single large (29Mb) file, chembox.ttl.

# 5. Stability/Reliability Evaluation

## 5.1. Introduction

In order to help users preserve and conserve their workflows first we have to think about why they break. Re-execute and reproduce the same results of a workflow over time is a hard task because of the volatility of some of their resources. The stability and reliability metrics aim to keep track and measure changes of completeness on a Research Object over time. Understanding workflow decay has needed an empirical study of workflow results after executing manually a sample collection of Taverna workflows. Although we focused on workflows for a specific system, our analysis can be applied to many other workflow design and execution environments. Thanks to this work we have been able to detect concrete causes on workflow decay and to create a classification of these causes. We used three criteria to select a representative group of workflows from myExperiment platform 1) year of creation 2) author and 3) domain of the workflow. In that way we covered workflows from different authors on 18 different scientific domains released between 2007 and 2012. Our analysis showed that approximately the 80% of the workflows failed in their execution, classified into the following categories:

- Volatile third-party resources: Web services or databases could be not available or accessible anymore or their functionality could have changed.
- Missing example data: Missing inputs, outputs or provenance information that could help to reproduce the workflow.
- Missing execution environment: Local servers, libraries or other missing software.
- Insufficient descriptions about workflows: Additional descriptions needed to understand the purpose of the workflow and its outputs.

After this study we discovered that the most repeated causes are the third party resources (50% of the workflows suffer this type of decay). However these results could vary with a different or bigger corpus.

While the completeness evaluation (which has been defined and explained in previous sections) allows us identifying the different types of decay by running it against a Research Object the reliability and stability metric (which are explained in this section)

add a new dimension: time. Thanks to time we can track the completeness over time in order to create a new metric that provides a value that reflects how much the user should trust a Research Object for reuse.

Stability measures the ability of a workflow to preserve its overall completeness state throughout a given time period. The stability measure is combined with the completeness measure to compute the reliability of a workflow. Stability extends the time scope of completeness taking into account the decay that the Research Object has suffered throughout its history. We understand reliability as a measure of confidence that the scientists can have in a particular workflow to preserve its capability to execute correctly and produce the expected results. Therefore, we propose completeness and stability as key dimensions to evaluate workflow reliability.

## 5.2. Completeness Assessment

The completeness dimension evaluates the extent to which a workflow satisfies a number of requirements specified in the form of a checklist following the Minim OWL ontology. Such requirements can be of two main types: compulsory (must) or recommendable (should). In order to be runnable and reproducible all the must requirements associated to a workflow need to be satisfied while should requirements propose a more relaxed kind of constraint. An example of the former is that all the web services invoked by the workflow be available and accessible (two of the main causes of workflow decay), while the presence of user annotations describing the experiment would illustrate the former. Since must requirements have a strong impact in the quality we have defined two thresholds: a) a lower bound $\beta_l$ which establishes the maximum value that the completeness score can have in case it does not satisfy all must requirements, and b) an upper bound $\beta_u$ which establishes the maximum value that the completeness score can have given that it satisfies all should and must requirements. Both $\beta_l$ and $\beta_u$ are parameterizable and can be configured on a case by case basis.

Therefore if at least a must requirement fails the completeness score is in the lower band [0-$\beta_l$] and otherwise in the upper band [$\beta_l$ - $\beta_u$]. Once identified the band, we define a normalized value of the completeness score as:

$$completeness\_score(RO,t) = f(RO_{(t)}, requirements, type) = \alpha \frac{nSReq(RO_{(t)}, must)}{nReq(must)} + (1-\alpha)\frac{nSReq(RO_{(t)}, should)}{nReq(should)} \in [0,1],$$

where t is the point in time considered, RO the research object that contains the workflow being evaluated, requirements the specific set of requirements defined within the RO for a specific purpose, type ∈ {must, should} the category of the requirement, α ∈ [0,1] is a control value to weight the different types of requirements, nSReq the number of satisfied requirements, and nReq the total number of requirements for the specified type.

### 5.3. Stability Assessment

The stability of a workflow measures the ability of a workflow to preserve its properties through time. The evaluation of this dimension provides the needed information to scientists like Bob the astronomer in order to know how stable the workflow has been in the past in terms of completeness fluctuation and therefore to gain some insight as to how predictable its behavior can be in the near future. We define the stability score as follows:

$$stability\_score(RO,t) = 1 - std(completeness\_score(RO, \Delta t)) \in [0.5, 1],$$

where completeness score is the measurement of completeness in time t and Δt is the period of time before t used for evaluation of the standard deviation.

The stability score has the following properties:

- It reaches its minimum value when there are severe changes over the resources of a workflow for the period of time Δt, meaning that the completeness score is continuously switching from its minimum value of zero (bad completeness) to its maximum of one (good completeness). This minimum value is therefore associated to unstable workflows.
- It has its maximum value when there are not any changes over a period of time Δt, meaning that the completeness score does not change over that time period. This maximum value is therefore associated to stable workflows.

- Its convergence means that the future behavior of the workflow can be predictable and therefore potentially reusable by interested scientists.

## 5.4. Reliability Assessment

The reliability of a workflow measures its ability for converging towards a scenario free of decay, i.e. complete and stable through time. Therefore, we combine both measures completeness and stability in order to provide some insight into the behavior of the workflow and its expected reliability in the future. We define the reliability score as:

$$reliability\_score(RO,t) = completeness\_score(RO,t) * stability\_score(RO,t) \in [0,1],$$

where RO is the research object, and t the current time under study. The reliability score has the following properties:

- It has a minimum value of 0 when the completeness score is also minimum.
- It has a maximum value of 1 when the completeness score is maximum and the RO has been stable during the period of time $\Delta$t
- A high value of the measure is desirable, meaning that the completeness is high and also that it is stable and hence predictable.

## 5.5. Implementation and integration

The implementation and integration of stability and reliability metrics in the context of Wf4ever have as a goal the interaction with the data available in the platform while providing useful APIs that offer both users and client applications access to structured traces of evaluations over time.

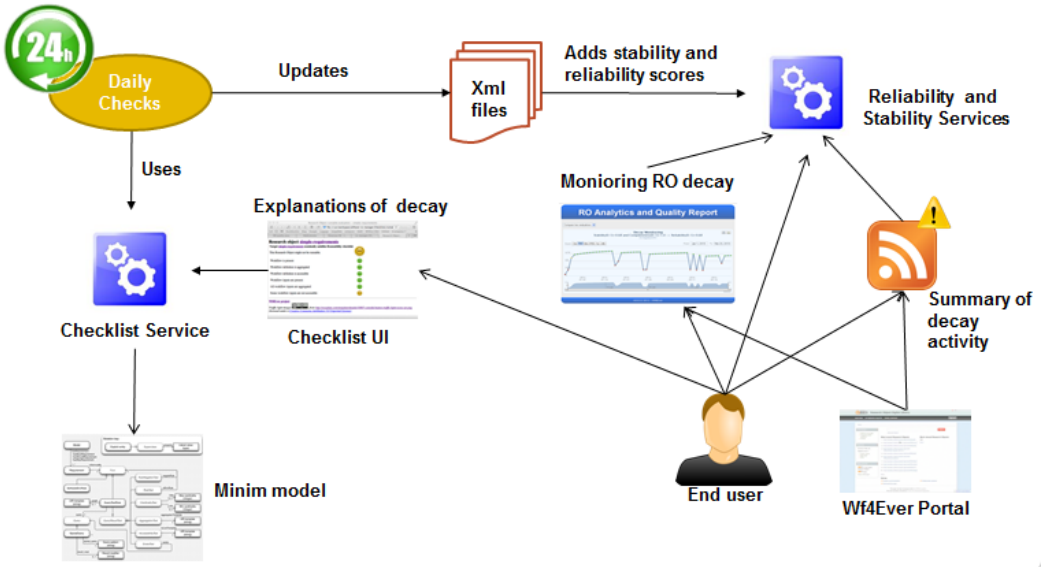



Figure 5 Wf4Ever quality assessment components interactions

In order to get the stability values of a Research Object we have to check their completeness values periodically over time. Once we have a completeness value for each point in time we are able to calculate the stability trace of the Research Object. This process is formed by various steps. First of all we have to identify all the existing Research Objects in RODL. To do so we perform a SPARQL query on the RODL endpoint which allows us to retrieve the URI that identifies every Research Object and we store them in a list. Iterating over the list of URIs gives us the target of the Checklist Evaluation service. The other parameters needed for the call of the checklist evaluation service such as the minim file and the purpose have been previously defined. After getting the results of the checklist evaluation service in JSON format we parse them in order to synthesize the content to the minimum needed for our calculations. We store all the rules together with their “pass” or “not pass” value and their level (must, should or may) in an xml file. Each Research Object has its own xml file that gathers the summarized evaluations. If the RO is evaluated for the first time we generate the xml file and store the summarized data. However, if the xml file had been created before then we edit it by adding the new evaluation.

Most of times, the evaluations do not change from day to another so we only need to store the evaluations that are different to their preceding. In that sense we are able to save a lot of memory. When the complete trace is requested by the one of the services it will be reconstructed by filling in the empty days based on the available data stored in the xml.

Once we have the full trace of completeness values we can proceed to calculate the stability trace (one stability value each day).

On the other hand we also have to calculate the reliability evaluation for the Research Objects. After having the trace with stability information together with the completeness value we can combine those to get the reliability trace. All this trace with the three values (completeness, stability and reliability) is going to be part of the response of the reliability REST web service. The remaining part of the response is formed by the rules of the checklist evaluations at each point in time. This kind of information helps the user to identify what happened at a specific point where the reliability and other values had decreased or improved. The complete set of results can be retrieved in both JSON and XML formats.

A different way to get the results is via notifications. Notifications provide a short summary of completeness, stability and reliability for the all days where completeness had changed on a specific time period requested by the user. The notification format follow the ATOM standard and users can subscribe to these notifications in order to get alerted when something happens with a Research Object of their interest (e.g. I want to get notifications for all my Research Objects so if a kind of decay affects them I will know and try to fix them).

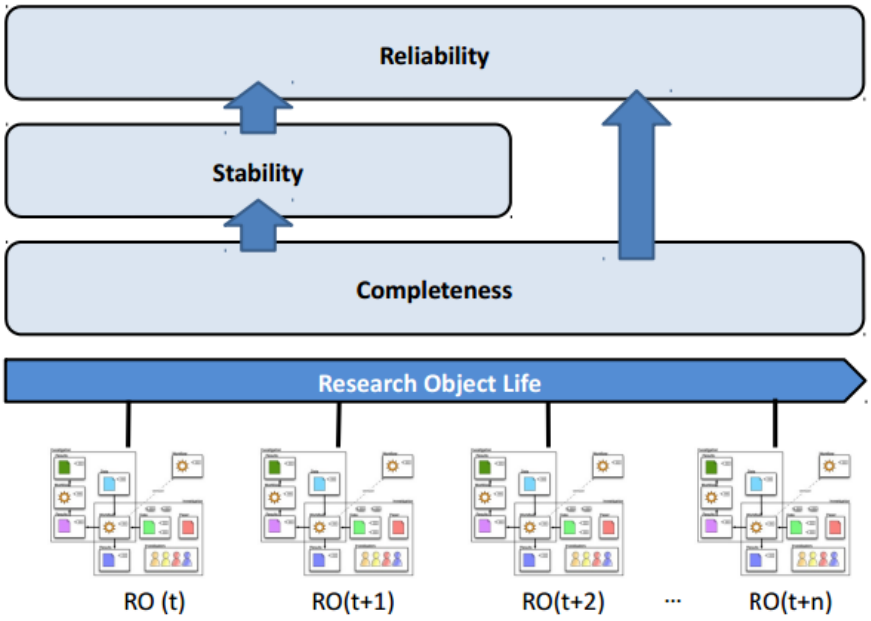



Figure 6 Layered Components of Reliability Measurement

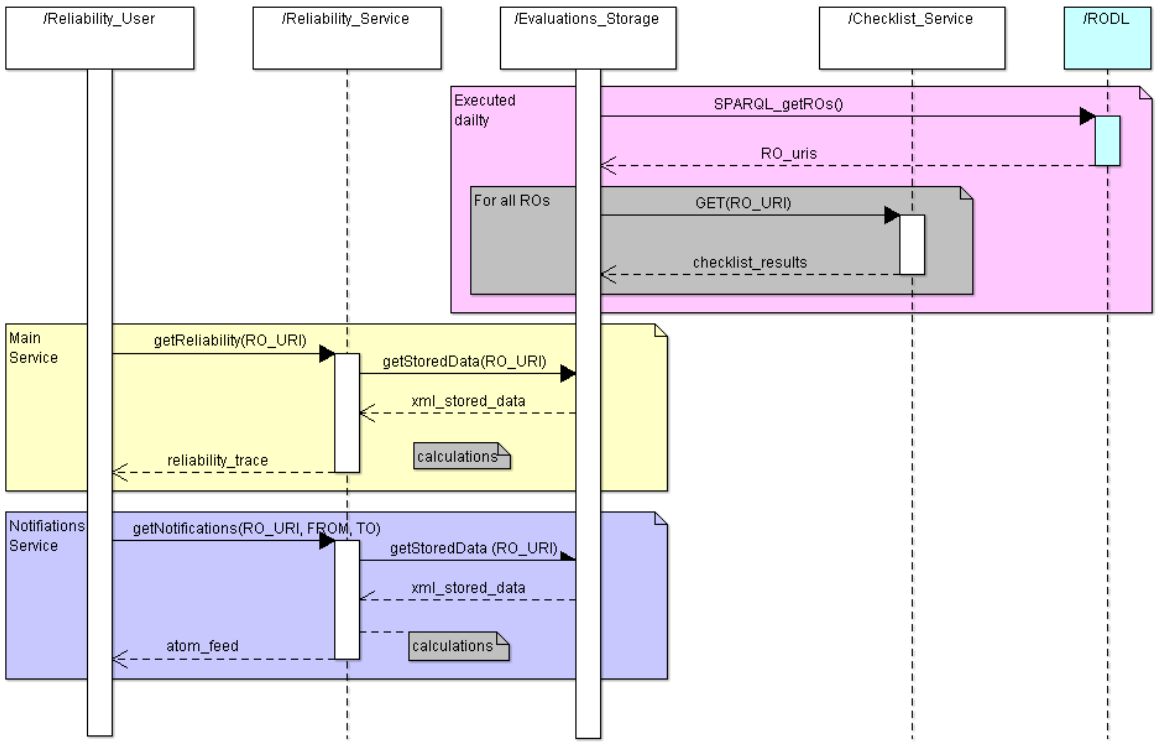



Figure 7 Sequence diagram for reliability evaluation, access, and notification services.

### Data Format

The data provided by the different quality criteria explained in this section is stored in different formats for its used and later accessibility.

- XML evaluation format (see Figure 8): the results of evaluating the accomplishment of the different defined RO requirements. This information is used for calculating stability and reliability scores. These files are then stored in the server side of the architecture.
- Stability and realiability format: the results of the stabThe way to call service is: http://sandbox.wf4ever-project.org/decayMonitoring/rest/getReliability?RO=roUri
  And depending on the accept header used the service will provide a xml or a json:

```
<trace>
<rouri>http://www…../ROs/ROid</rouri>
<evaluations>
<eval evalresultclass="must">
<date>2013,5,9,15,17</date>
<checklistitems>
       <checklistitem itemlevel="must" itemsatisfied="true">Third party resources
accessible</checklistitem>
       <checklistitem itemlevel="must" itemsatisfied="true">Third party resources have not
changed</checklistitem>
       <checklistitem itemlevel="should" itemsatisfied="true">Execution environment
available</checklistitem>
       <checklistitem itemlevel="should" itemsatisfied="false">Workflow description not
available</checklistitem>
</checklistitems>
</eval>
</evaluations>
</trace>
```

Figure 8 Evaluation of a Research Object presented in XML format.

## 5.6. Presentation of data: RO-Monitoring Tool

The monitoring tool provides a visual and friendly way to explore daily evaluations of completeness alongside with its correspondent values of stability and reliability that provide a more comprehensive way to get the data. The monitoring tool offers all the information related to reliability and evaluations for a specific research object. The graph covers time on the X axis and reliability on the Y axis. Each point in time can be clicked to get the set of rules that were evaluated and their results for that point in time. The monitoring tool application is available at: http://sandbox.wf4ever-project.org/decayMonitoring/visual.html?id=rouri

```
<itemReliability>
<rouri>http://www.…./ROs/ROid</rouri>
<completeness>0.733</completeness>
<stability>0.9059206882491023</stability>
<reliability>0.664039864486592</reliability>
<evaluation>
        <date>2012,4,16,12,33</date>
        <evalresultclass>pass</evalresultclass>
        <completeness>1.0</completeness>
        <stability>1.0</stability>
        <reliability>1.0</reliability>
        <checklistitems>
                <itemlevel>must</itemlevel>
                <itemsatisfied>true</itemsatisfied>
                <itemlabel>Third party resources available</itemlabel>
        </checklistitems>
</evaluation>
</itemReliability>
```

Figure 9 Stability and Reliability evaluation presented in XML format.

### Evaluation of monitoring tool

This section shows the toosl used for the monitorization and for the improvement of the user experience by providing to them visualization tools of the above introduced quality dimensions. This monitoring tool is integrated in the Wf4Ever Sandbox and is available at[6]. We also explain the implementation done towards the generation of that live demo and the APIs that we have specified for allowing the reuse of the services for any other purposes inside or outside of the project.

We have done a first evaluation of the monitoring tool to measure the potential benefit for a successful reuse of taking into account a historical perspective on the health of scientific workflows, represented by the reliability score, as opposed to instantaneous quality measures like the completeness value. We simulated a year of changes over a hundred workflows based on the data we obtained on the study of decay explained in

[6] http://sandbox.wf4ever-project.org/decayMonitoring/monitor.html

the introduction for reliability section. Those simulations were presented to a set of scientists that had access to the first 274 days of the history of simulations and based on that decide if they would reuse that Research Objects or not by answering two questions: 1.Would you reuse this workflow for your own experiments today?, and 2.Would you use it in three months from now?.

We compared the results of these questions with the full history of reliability against the same question after seeing only an isolated completeness evaluation in day 274.

Seeing our results we can say that their choice using the monitoring tool was 76% better than without it. We can confirm based on our evaluation that the use of reliability score improves the results obtained using completeness information exclusively. This shows evidence that the use of reliability information, based on the record of workflow health over time, enables scientists to make more informed and better decisions about the reuse of third party scientific workflows.

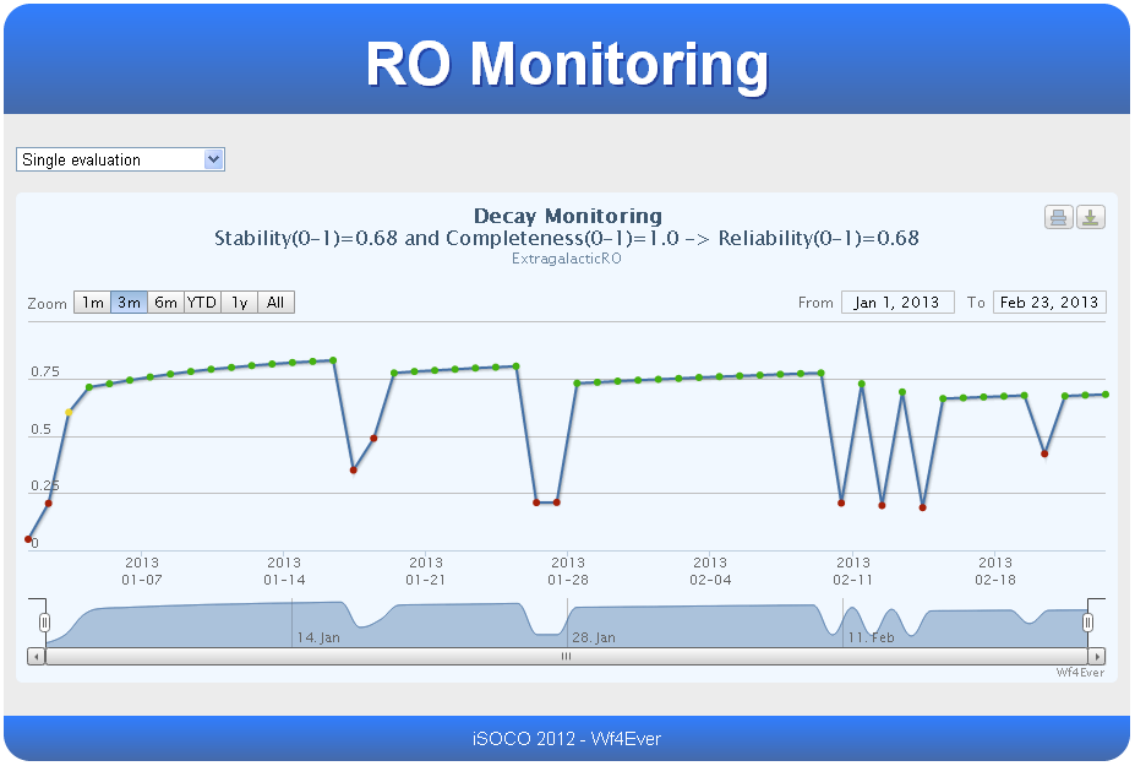



Figure 10 Wf4Ever RO-Monitoring Tool

# 6. Conclusions

## 7. Publications


- Khalid Belhajjame, Jun Zhao, Daniel Garijo, Aleix Garrido, Stian Soiland-Reyes and Pinar Alper, “The Taverna and Wings Workflow PROV-Corpus, International Workshop on Managing and Querying Provenance Data at Scale Held in conjunction with EDBT/ICDT 2013.
- Why Workflows Break - Understanding and Combating Decay in Taverna Workflows. Zhao J, Gómez-Pérez JM, Belhajjame K, Klyne G, García-Cuesta E, Garrido A, Hettne K, Roos M, De Roure D, Goble CA. 8th IEEE International Conference on e-Science (e-Science 2012).
- MIM: A Minimum Information Model Vocabulary and Framework for Scientific Linked Data. Gamble M, Goble CA, Klyne G, Zhao J. 8th IEEE International Conference on e-Science (e-Science 2012).
- José Manuel Gómez-Pérez, Esteban García-Cuesta, Jun Zhao, Aleix Garrido and José Enrique Ruiz, “How Reliable is Your workflow: Monitoring Decay in Scholarly Publications” Sepublica’2013 Workshop held jointly with the ESWC’2013 (Best Paper Award).
- Belhajjame, Khalid, Jun Zhao, Daniel Garijo, Aleix Garrido, Stian Soiland-Reyes, Pinar Alper, and Oscar Corcho. "A workflow PROV-corpus based on taverna and wings." In Proceedings of the Joint EDBT/ICDT 2013 Workshops, pp. 331-332. ACM, 2013.
- José Manuel Gómez-Pérez, Esteban García-Cuesta, Aleix Garrido and José Enrique Ruiz, "When History Matters - Assessing Reliability for the Reuse of Scientific Workflows", in-use track held at ISWC2013 21-25 October, Sydney, Australia.
- Zhao J, Klyne G. “RO-Manager: A Tool for Creating and Manipulating Research Objects to Support Reproducibility and Reuse in Sciences.”, Linked Science 2012 W3C Candidate Recommendation 11 December 2012 http://www.w3.org/TR/prov-o/
- Missier P, Belhajjame K. “A PROV encoding for provenance analysis using deductive rules.” Proceedings of IPAW, Springer, 2012
- Zhao, Jun; Sahoo, Satya S.; Missier, Paolo; Sheth, Amit; Goble, Carole. “Extending Semantic Provenance into the Web of Data.”, IEEE Internet Computing, vol.15, no.1, pp.40-48, Jan.-Feb.

## 8. References

[CHECKLIST]: http://www.ncbi.nlm.nih.gov/pubmed/16990087 (B. Hales and P. Pronovost, “The checklist-a tool for error management and performance improvement”, Journal of critical care, vol. 3, no. 21, pp. 231-235, 2006.)

[MIBBI]: http://www.nature.com/nbt/journal/v26/n8/pdf/nbt.1411.pdf (C. Taylor, D. Field, S. Sansone, J. A. R. Aerts, A. M., B. P. Ball C.A., M. Bogue and T. Booth, “Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the MIBBI project”, Nature biotechnology, vol. 8, no. 26, pp. 889-896, 2008.)

[MIM]: http://dx.doi.org/10.1109/eScience.2012.6404489 (Matthew Gamble, Jun Zhao, Graham Klyne, Carole Goble. "MIM: A Minimum Information Model Vocabulary and Framework for Scientific Linked Data", IEEE eScience 2012 Chicago, USA October, 2012)

[MIM-spec]: http://purl.org/net/mim/ns (Minimum Information Model Vocabulary Specification)

[Minim-OWL]: http://purl.org/minim/ (Minim ontology)

[Minim-results]: http://purl.org/minim/results (Model for Minim-based checklist evaluation results)

[Minim-spec]: https://github.com/wf4ever/ro-manager/blob/develop/Minim/minim-revised.md (Minim checklist description)

[Minim-owldoc]: http://purl.org/minim/owldoc (Minim ontology OWLDoc documentation)

[D4.2v1]: http://repo.wf4ever-project.org/Content/39/D4.2v1Final.pdf (Esteban García-Cuesta (iSOCO), Jun Zhao (OXF), Graham Klyne (OXF), Aleix Garrido (iSOCO), Jose Manuel Gomez-Perez (iSOCO), “Design, implementation and deployment of Workflow Integrity and Authenticity Maintenance components – Phase I. Deliverable D4.2v1, Wf4Ever Project, 2012,” 2012.)

[D2.2v2]: (@@uri-tbd) (S. Bechhofer, Khalid Belhajjame, et. al., “Design, implementation and deployment of workflow lifecycle management components - Phase II. Deliverable D2.2v2, Wf4Ever Project, 2013,” 2013.)

[WF-decay]: http://users.ox.ac.uk/~oerc0033/preprints/why-decay.pdf (Zhao J, Gómez-Pérez JM, Belhajjame K, Klyne G, García-Cuesta E, Garrido A, Hettne K, Roos M, De Roure D, Goble CA, "Why Workflows Break - Understanding and Combating Decay in Taverna Workflows", 8th IEEE International Conference on e-Science (e-Science 2012).)

[Zhao'12]: J. Zhao, J.M. G´omez-P´erez, K. Belhajjame, G. Klyne, E. García-Cuesta, Garrido A, Hettne K, Roos M, De Roure D, Goble CA. Why Workflows Break - Understanding and Combating Decay in Taverna Workflows. In the proceedings of the IEEE eScience Conference (eScience 2012), IEEE CS, Chicago, USA, 2012.

[Cicca'11] P. Ciccarese, M. Ocana, L.J. Garcia Castro, S. Das, and T. Clark. An open annotation ontology for science on web 3.0. J Biomed Semantics, 2(Suppl 2):S4, 2011.