Why Workflows Break - Understanding and Combating Decay in Taverna Workflows

Jun Zhao*, Jose Manuel Gomez-Perez†, Khalid Belhajjame‡, Graham Klyne*, Esteban Garcia-Cuesta†
Aleix Garrido†, Kristina Hettne¶, Marco Roos¶, David De Roure‖, Carole Goble‡
*Department of Zoology, University of Oxford, Oxford, UK {jun.zhao, graham.klyne}@zoo.ox.ac.uk
†iSOCO, Madrid, Spain {jmgomez, egarcia, agarrido}@isoco.com
‡School of Computer Science, University of Manchester, Manchester, UK
khalidb@cs.manchester.ac.uk, carole.goble@manchester.ac.uk
§Leiden University Medical Centre, Leiden, NL {k.m.hettne, m.roos}@lumc.nl
‖Oxford e-Research Center, University of Oxford, Oxford, UK david.deroure@oerc.ox.ac.uk

***Abstract*—Workflows provide a popular means for preserving scientific methods by explicitly encoding their process. However, some of them are subject to a decay in their ability to be re-executed or reproduce the same results over time, largely due to the volatility of the resources required for workflow executions. This paper provides an analysis of the root causes of workflow decay based on an empirical study of a collection of Taverna workflows from the myExperiment repository. Although our analysis was based on a specific type of workflow, the outcomes and methodology should be applicable to workflows from other systems, at least those whose executions also rely largely on accessing third-party resources. Based on our understanding about decay we recommend a minimal set of auxiliary resources to be preserved together with the workflows as an aggregation object and provide a software tool for end-users to create such aggregations and to assess their completeness.**

I. Introduction

We have witnessed in the last 10 years an increased uptake of workflows as the technology of choice for computational scientific experiments [16]. A workflow experiment is composed of a set of coordinated computational tasks. Each task takes some data inputs and produces some data outputs, which are consumed by subsequent tasks according to the workflow definition. As well as providing the means for automating scientific experiments, workflows have their own scientific value: they capture experimental methods designed by the scientists to test a given hypothesis, confirm a known fact, amongst other things. Hence workflows play an important role as a medium for sharing, exchanging and reusing experimental methods, as demonstrated by existing workflow repositories, such as myExperiment [14] and crowdLabs [27].

Resources required for executing workflows, like services and data, can be either local and hosted along with the workflow or remote, such as public repositories or web services hosted by third parties. Over time these workflows, particularly those from the life sciences domain, are notably subject to a decayed or reduced ability to be executed or produce the same results [32]. This is what we call *workflow decay*. Scientific communities cannot fully benefit from the potential of computational workflows until the problem of workflow decay is addressed. Our analysis with the myExperiment workflow repository shows that a good proportion of its workflows do suffer from decay, which decreases their value. But *what are the causes of workflow decay, and how can we detect, prevent and cure it?*

To address the above questions, we manually re-executed a sample collection of workflows written for the widely used Taverna workflow workbench [31] from the myExperiment repository in order to identify the causes of their decay. Taverna workflows are a good example for this study because they are particularly vulnerable to decay due to third party dependencies. Although our analysis was based on a specific workflow system, the results and methodology should be applicable to workflows from other systems, many of which commonly require access to third-party resources, or sufficient documentation for successful execution. We identify the main causes of workflow decay and the impact they have on workflow execution or the reproducibility of results, and we draw a classification of these decay causes.

Our analysis of decay also shows that a minimal set of information elements can be preserved together with workflows in order to reduce their decay over time. To test our understanding of combating workflow decay, we have implemented software tooling (RO-Manager) to construct and evaluate aggregations of workflows and the minimal set of auxiliary information. The resulting aggregations are represented as **Research Objects** (RO) [5].

In the rest of the paper we begin by reviewing existing proposals in Section II. We then motivate the need for addressing the problem of workflow decay in Section III and present our analysis and classification of the causes of decay in Section IV. We show that it is possible to identify a minimal set of information to be aggregated with a workflow to address some of the problems of decay. We present an outline of our approach in Section V and our implementation in Section VI. We validate our solution by a case study (in Section VII). Finally, Section VIII concludes the paper by outlining our main contributions and ongoing work.

II. Related Work

Our discussion covers two areas: approaches for repairing workflow decay and modelling aggregation structure.

The problem of workflow decay is recogised as an impediment to the reuse of workflows and the reproducibility of their results [16]. Yet, we observe only a handful of approaches attempting to address the problem of workflow decay, with most of them focusing on *repairing* the decay. In this paper, we do not aim to present yet another technique for repairing workflow decay, rather we aim to identify the root causes that may yield decay in a workflow, and propose the elements that if combined with the workflow will facilitate its repair or prevent its decay in the future. Information used by the existing repair techniques, such as provenance information, could provide valuable inputs to extend our proposal. The existing repair approaches can be classified into the following three categories:

- *Repair by service substitution*: Our empirical study shows that a lot of workflow decay is caused by the unavailability of third-party web services that are required for the execution of workflows. Therefore, an effective repair technique could be replacing unavailable services with suitable substitutes. This has been investigated in our previous work [4], [6], in which semantic annotations of web services and/or provenance traces of previous executions of workflows were exploited to identify suitable substitute services. Similarly, Calore *et al.* [9] proposed using string-based similarity measures to search for substitute web services by comparing their annotations with those of the unavailable web services.
- *Repair by adaptation*: Lee *et al.* [25] proposed adaptation techniques that can be applied to react to changes in the execution environment of the workflow. Although their original objective is to improve the performance of workflow, the techniques are potentially applicable to the problem of workflow decay. The workflow abstraction layer proposed by Gil *et al.* [16] also provides the desired flexibility for workflows to adapt to changes.
- *Repair by provenance support*: Sometimes unavailable services can neither be substituted nor repaired. Provenance information can be useful to explain failures and to enable appropriate repairs, by tracing what changes to the components of a workflow led to this incapability of replicating [10], [13], [23]. Additionally, provenance information can also be used to run the workflow using previous states of the underlying data sources, thereby guaranteeing the reproducibility of the workflow results [24]

Approaches like Virtual Machines have also been exploited [34] to preserve runtime environment to cope with changes. However, they do not currently support preservation of the distributed environment in which execution occurs, such as client-server models or calling remote services which may be third party. Workflows share many properties with software, such as the composition of components with external dependencies. Hence some aspects of software preservation [28] are applicable and should be further investigated.

Research Objects were conceived to extend traditional publication mechanisms [3]) and take us "beyond the pdf" [7] by aggregating essential resources related to experiment results along with publications. This includes not only the data used but also methods applied to produce and analyse that data. The notion of using aggregation to promote reproducibility and accessibility of research has been studied in many other work, including the Open Archives Initiation Object Reuse and Exchange Specification (OAI-ORE, or ORE [1]), the Scientific Publication Packages (SPP) [22], and the Scientific Knowledge Objects [18]. Nano-publication [20] is another approach of supporting accessible research by publishing key results of an investigation as succinct, tripartite statements. The RO model has a workflow-centric dimension, providing modules to describe a workflow experiment (such as the processors used or the data parameters processed) as well as provenance traces of its executions. These workflow-centric ROs are aimed to support particularly the reproducibility and reuse of workflow-based experiments [19]. This is closely related to packs from myExperiment [30], which aggregate elements such as workflows, documents and datasets together, following Web 2.0 and Linked Data principles. The RO model is built upon myExperiment packs, but it has a more dedicated representation of workflows and provenance.

In order to enhance the trustworthiness of these ROs we associate them with a list of explicitly defined requirements that they must satisfy and we use this list to evaluate their completeness, i.e. the quality of the ROs. This is built upon the idea of the (Minimum Information Model) MIM model by Gamble *et al.* [15], which provides an encoding of these requirements using the standard Web Ontology Language (OWL)[1] and makes use of existing OWL reasoning software to validate data against these checklists.

III. Motivation: The Decay of A Workflow

To illustrate the needs for preserving scientific workflows, we use an example workflow which is part of the workflow from a myExperiment pack[2]. The workflow is used to extract a list of common Kyoto Encyclopedia of Genes and Genomes (KEGG) pathways[3] for genes from two related studies. It is illustrated in Figure 1. The workflow starts by issuing, in parallel, two queries to the KEGG database: the first retrieves pathways found for genes in a special chromosome previously identified being related to a studied disease, and the second extracts pathways for genes differentially expressed in a local microarray study. The pathways returned from the two searches are then compared to identify common pathways by a locally hosted web service. By identifying pathways that are common to both studies, a more informative picture can be obtained of the candidate processes involved in the expression of a phenotype.

[1]http://www.w3.org/2004/OWL/
[2]http://www.myexperiment.org/packs/55.html
[3]http://www.genome.jp/kegg/

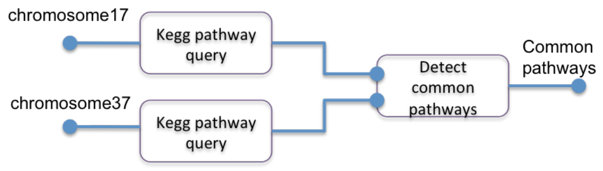

Fig. 1. Identifying intersecting pathways.

The content of the KEGG database is subject to regular updates, and the implementation of the web service for identifying common pathways was locally hosted by the PhD student who initially performed the study. Thus it is possible, and likely, that when we re-ran the workflow it would produce different lists of pathways. Should the KEGG database alter its interface then the workflow might no longer be able to access it and it will become inoperable, and there is no guarantee of continued availability of the web service hosted by the PhD student.

Such volatility of third party resources, including the databases, web services, and persons involved in the experiments, lead to the decay of workflows, which is a fundamental challenge for preserving their reusability and reproducibility. The symptoms are common but the consequences are severe. Workflows are perceived as an effective preservation of scientific methods by transparently documenting the process of an experiment. A failure to ensure their longevity will impair scientists' trust in results whose methods can no longer be run or reproduced.

The above analysis shows that to combat decay it is not enough to just preserve the workflow; we must also make a suite of auxiliary information available together with the workflow. To know what these auxiliary information could be, we must have a more comprehensive understanding about the causes of the decay. Therefore, we took a bottom-up approach by analysing a selection of myExperiment workflows to gather actual possible causes of their failure, as presented in the following section.

IV. A Classification of Causes to Workflow Decay

In this section we present our analysis of concrete causes of workflow decay and a classification of these causes[4]. We chose Taverna workflows because this is the largest available workflow collection (more than half of the workflows in myExperiment are Taverna workflows at the time of writing) and Taverna workflows have been published on myExperiment since its launch in 2007, therefore providing a good insight into decay over those years. Although in this paper we focus on a particular family of workflows, we expect our approach and analysis to be applicable to many others and our analysis to be repeatable on a different corpus of workflows.

92 Taverna workflows were chosen from myExperiment, including an equal number of Taverna 1 [31] and Taverna 2 [29] workflows. To base our analysis on a sample of workflows that is representative of the set of workflows in myExperiment, we tried to select workflows by three criteria: 1) the year they were created, 2) the creator of the workflows; and 3) the domain studied by the workflows. Figures 3-6 in Appendix IX illustrate the features of our workflow corpus. We believe that the decay of workflow could be directly impacted by the year they were created, hence we tried to make an even coverage of T1/T2 workflows between the years 2007 and 2012. In order to reduce possible bias introduced by the specific workflow creators, we avoided choosing workflows created by the same person in the same year. Our workflow selection also had a good coverage of domains, covering 18 different scientific (such as life sciences, astronomy, or cheminformatics) and non-scientific domains (such as testing of Grid services).

Some previous work has performed analysis of the whole myExperiment repository in order to understand the usage patterns of workflows [26], [35]. A complete analysis of myExperiment repository is not the goal of our study, rather we focused on understanding the concrete causes of workflow decay by choosing a sufficiently representative corpus from myExperiment.

To identify the causes of decay that these workflows may suffer from, we attempted to execute them using the Taverna 2.3 workbench. We then manually examined their results, diagnosed broken links, etc. Our analysis showed that nearly 80% of the tested workflows failed to be either executed or produce the same results (if testable), and those from earlier years (2007-2009) had more than 80% failure rate (as shown in our Figures 7-8 in Appendix IX). The causes of workflow decay can be classified into four categories, as presented in the following.

A. *Volatile third-party Resources*

Most of the workflows that we analysed make use of third-party resources such as web services and databases, e.g., the KEGG services used in our example workflow provided by the Data Bank of Japan. The provision of such resources may be interrupted or changed, causing failure of the workflow to execute. In certain cases, the workflow cannot be run, even when the third party resources that it relies on are available, e.g., when such resources require authentication. Another cause that may lead to workflow decay, is changes to third party resources. For example, if the web service provider decides to change the implementation of the web service, then the workflow execution may not deliver the same results, or worse, it may not be possible to execute . Table I summarises these causes of decay with concrete examples.

B. *Missing example data*

It is not always obvious which data can be used as inputs to the workflow execution, and example inputs are often most helpful. Example outputs can also be useful to gain an insight of the outcome anticipated from the workflow. However, our analysis revealed that they are not always made available. Provenance traces of previous runs are also useful as indications of where example data may be found.

[4]All related experiment data can be downloaded in our research object that is available at http://purl.org/NET/wf4ever/ro/ro-decay-paper.

TABLE I
CATEGORISATION OF DECAY CAUSED BY THIRD-PARTY RESOURCES

Causes	Refined causes	Examples
Third party resources are not available	Underlying dataset, particularly those locally hosted in-house dataset, is no longer available	Researcher hosting the data changed institution, server is no longer available
	Services are deprecated	(DNA Data Bank of Japan) DDBJ web services are not longer provided despite the fact that they are used in many myExperiment workflows
Third party resources are available but not accessible	Data is available but identified using different IDs that the one known to the user	Due to scalability reasons the input data is superseded by new one making the workflow not executable or providing wrong results
	Data is available but permission, certificate, or network to access it is needed	Cannot get the input, which is a security token that can only be obtained by a registered user of ChemiSpider
	Services are available but need permission, certificate, or network to access and invoke them	The security policies of the execution framework are updated due to new hosting institution rules
Third party resources have changed	Services are still available by using the same identifiers but their functionality have changed	The web services are updated intentionally or unintentionally (e.g.malware) providing wrong results

C. Missing execution environment

The execution of a workflow may rely on a particular local execution environment, for example, a local R server or a specific version of workflow execution software. Some of our test workflows exhibit this type of decay. Taverna often provides sufficient information about missing libraries, and sometimes workflow descriptions provide a warning about the requirement for a specific library. This type of decay appears to be fixable by installing the missing software, albeit requiring some effort.

D. Insufficient descriptions about workflows

Sometimes a workflow workbench cannot provide sufficient information about what caused the failure of a workflow run. Additional descriptions in the workflow can play an important role in assisting re-users to understand the purpose of the workflow and its expected outcomes.

E. Summary

The results of our analysis are summarised in Figure 9 in Appendix IV, which illustrates the number of workflows that suffer from each of the causes of decay presented above. It shows that 50% workflows suffer from decay due to third party resources. However, we might draw a different conclusion if a bigger corpus or a different collection of workflows are used. Examining the causes of decay due to third party resources, we observed the unavailability of third party resources as the leading cause, followed by their inaccessibility, and service changes (see Figure 10 in Appendix IV).

Informed by this analysis, Table II illustrates the set of minimal information that we believe should be preserved along with a workflow for two specific aspects of combating workflow decay: 1) guaranteeing the rerunning of a workflow; and 2) assessing the ability of a workflow to replicate previous results. This table was created based on our hands-on experience of attempting to repair failed workflows during our empirical study. Different approaches might require a different set of information from our proposal, but the structure of the table lays out a landscape for any future investigation.

TABLE II
INFORMATION ELEMENTS REQUIRED FOR SPECIFIC TASKS OF COMBATING WORKFLOW DECAY.

Causes	Increase runnability	Assess replicability
Missing third-party resources	example inputs+local copy of third-party resources	example inputs+outputs+ local copy of third-party resources
Inaccessible third-party resources	example inputs+auxiliary descriptions about key vulnerable resources	example inputs+outputs+ local copy of third-party resources
Updates of third-party resources	example inputs	example inputs+outputs+ snapshots of third-party resources
Missing example data	example inputs	example inputs and outputs
Missing execution environment	example inputs+necessary execution libraries	example inputs+outputs+ necessary execution libraries
Incomplete metadata	example inputs+detailed metadata descriptions	example inputs+outputs+ detailed metadata descriptions

From the table we identify that although a different set of information could be required for the two specific tasks, a small set of *common* information can provide a minimal starting-point. For example, to assess the replicability of a workflow, we must at least preserve the information about previous outputs to assess whether replication was indeed achieved. We could recommend to scientists that these minimal sets of information be preserved along with their workflows, in order to increase their repeatability and reproducibility.

Furthermore, Table II also suggests a need to support customising this minimal list of requirements for specific workflows that might be associated with components subject to different decay causes. For example, we identify that although example inputs are commonly required for running a workflow, snapshots of third-party services or databases can also be required by some workflows to replicate previous results.

V. APPROACH

Based on our analysis we apply an approach of bundling the preserved workflow together with the auxiliary information for mitigating its decay. The resulting aggregation is a *Research Object*. How much supporting information is enough information? It may be that one can never have enough information, which gives rise to a notion of *completeness* that must be considered with respect to some purpose or goal.

Our decay analysis identifies a minimal set of information to support the goal of reducing workflow decay. To confirm that an RO is complete in its provision of this information we adopt a *checklist-based* approach. Checklists are a well-established tool for guiding practices to ensure safety, quality and consistency in the conduct of complex operations [11],

[21]. More recently, they have been adopted by the biological research community to promote consistency and comparability across research datasets [33], in the form of minimum information guidelines. Here we apply checklists to review and quality evaluation of ROs. In the following we introduce the RO model used to represent the aggregation object, and the Minim model for representing the minimal requirement checklists.

A. *RO Model and Manifest*

The RO model is an aggregation structure for collecting information about the mechanics and context of an experiment, including workflow description, data and supporting information, in order to facilitate the reuse and reproducibility of the experiment [3]. The structure of an RO is described in a *manifest* file. A manifest file to describe an RO for our example workflow (Figure 1) includes the following information:

- References to the various resources aggregated by the RO, e.g., the workflow file and the input and output data used in its previous run. These aggregated resources can either be resources on the web that are referred to in the manifest file or stored physically within an RO.
- Annotations about the RO and its aggregated resources, such as the creator of a workflow, when it was created, or indication of a specific library required to execute the workflow. These annotations are mainly structured information, expressed using the W3C standard Resource Description Framework (RDF)[5] data format.

The RO model is implemented as a suite of lightweight ontologies using OWL, and makes use of existing community vocabularies. Its aggregation structure is based on the Object Reuse and Exchange (ORE) vocabulary [1] and its expression of annotations to its aggregated resources is based on the Annotation Ontology (AO) [12].

B. *Checklists for RO completeness*

To evaluate the completeness of an RO we need to express them in a way to allow for machine evaluation and reporting. We use the *Minim* data model[6], which is a customised version of the aforementioned MIM model.

The MIM model [15] is inspired by the Minimum Information for Biological and Biomedical Information (MIBBI)-style minimum information models [33]. It represents the requirements that the data is required to satisfy in a knowledge representation language (OWL) and uses information about the data provided in a machine-readable format (RDF). Evaluation of the conformance of some test data against the MIM requirements is performed using existing OWL reasoning software.

The Minim model used in our approach is an evolution of the MIM model that trades the theoretical elegance of evaluation using OWL reasoning for flexibility to introduce *ad hoc* testing capabilities that are not limited to examination of available RDF data, and which can, in principle, invoke arbitrary probes of local and web-accessible resources. Figure 2 provides a schematic view of the Minim model. As with MIM, a Minim checklist enumerates a set of requirements to be satisfied. It differs from MIM in the following key respects:

- An explicit expression of *Constraints* (or Goals): 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. Constraints are a basis for selecting a model (checklist) to use, corresponding to the purpose of a given evaluation.
- An extended expression of *Requirements*: these follow the MIM idea, but are extended to cover more than just the presence of certain information about an experiment. 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).
- An introduction of *Rules*: for requirements that go beyond testing for availability of certain information, we replace the OWL-based definition of *reports* with a system of rules, each of which invokes a *checklist primitive* testing operation (see Section VI).

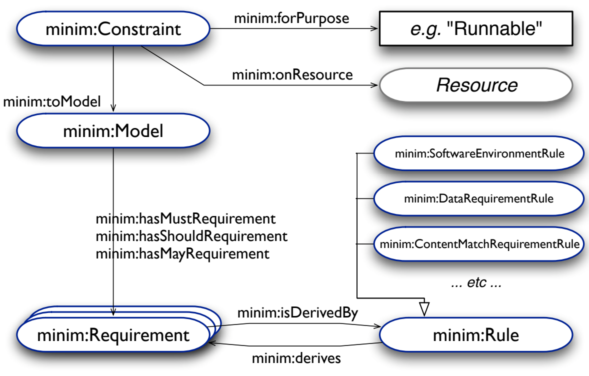

Fig. 2. An overview of the Minim model.

VI. Implementation

In this section we describe our implementation of the RO model and Minim checklist evaluation in a user-facing command-line tool, RO-Manager.

A. *RO-Manager*

RO-Manager[7] is a command-line tool that supports the creation, manipulation and evaluation of an RO in a local file system directory structure. It supports bundling of information into an RO to aid workflow preservation, and evaluating the completeness of the RO against a checklist. The evaluation component has also been implemented as a REST service API.

B. *Checklist evaluation*

The evaluation component can evaluate an RO against a specified checklist, taking account of the state of any web resources referenced, and provide an indication of whether the RO is (still) fit for purpose. The tool takes an *RO*, including its manifest, a *Minim description* of one or more *checklist*s, and a *purpose* that the RO is assessed against. The indicated purpose

[5]http://www.w3.org/RDF/
[6]http://purl.org/minim/
[7]RO-Manager is on Github: https://github.com/wf4ever/ro-manager

is used to select a checklist from those available, against which the RO is evaluated to produce an *evaluation report*. Here is a simple example checklist expressed using RDF Turtle[8] format following the Minim model:

```
<#runnable_RO_model> a minim:Model ;
  rdfs:label "Runnable RO" ;
  minim:hasMustRequirement
    <#isPresent/workflow-instance>,
    <#isPresent/workflow-inputfiles> ,
    <#isPresent/workflow-inputportnames> .
```

For our KEGG pathway example (Figure 1), this amounts to requiring the following *must* be present, which are the minimal set of information required for the workflow to be runnable:

- A Taverna workflow description file;
- All the input files (chromosome name, start position, end position) required for the gene search;
- Workflow process port names for each of the input files.

Each of these requirements are actually derived (reported) through invocation of a *checklist primitive*, described in the following section.

C. Checklist requirements and primitives

The checklist entries are high-level requirements to be satisfied, each of which invokes a *checklist primitive* test. So far we have identified and implemented the following high level requirements:

- *Evaluate the presence of a particular type of resource*: e.g., is a particular type of resource (e.g. a workflow description) aggregated by the RO? Are all the input values present? Is provenance information about a workflow run available?
- *Evaluate the accessibility of a particular type of resource*: e.g., are all workflow descriptions accessible? Are any input files associated with a workflow actually accessible?
- *Evaluate of the presence of dependency between ROs*: e.g., does one RO have a dependency on another RO? Does an RO depend on another RO that does not meet some quality criteria? Does an RO depend on another RO whose publication has been retracted?

Each of the above requirements can be implemented as one or more checklist primitives. The choice of checklist primitives is intended to allow a range of requirement tests to be performed using a relatively small number of re-purposable primitives. New primitives can be added as required by modifying the RO Manager evaluation software. So far the following primitives have been designed and implemented based on the above checklist functions:

- assess the presence of a particular resource in an RO, using SPARQL ASK and/or SELECT queries, e.g. are any scripts or programs used by a workflow aggregated by an RO?.
- assess the liveness (accessibility) of a resource aggregated in an RO. This can test for existence of a local file, or accessibility of an external web resource, depending on the form of reference used in the RO; e.g., are the inputs for a workflow instance available?
- assess the availability of a certain program in the software environment by executing a command and matching the response against a supplied regular expression; e.g., is Python version 2.6 or greater installed?

VII. Preserving workflows in practice

The main goal of our experiment is to assess the effectiveness of our checklist and minim approach to reduce workflow decay. For this we used a collection of ROs that we created using existing myExperiment packs. Specifically, we used the RO Manager tool to create the ROs from myExperiment packs and its checklist component to assess their completeness for two specific purposes: **workflow runnability** and **result replicability**.

Note that there is a sizable number of packs in myExperiment. However, our previous investigation shows that only a small proportion of them are actually created for preserving workflows from decay[9]; the rest are mainly used for either sharing knowledge, by creating a collection of documents and papers, or for self-curation, with little information to make the workflow (re-)usable. Based on this investigation we identified, for our analyses, 4 packs (pack 55, 58, 217, and 219), from the $\sim$100 Taverna workflows used in our decay study, as our candidate ROs, described in Table III.

The transformation, whereby a myExperiment pack was converted into an RO, was implemented by the RO Manager tool. The purpose of the transformation was not to produce a complete mapping between myExperiment packs and the RO model, which is out of the scope of this work. On the contrary, we transformed just-enough content into the corresponding RO representation for our evaluation, and added additional files required for our completeness evaluation.

A. Workflow runnability

According to the analysis result shown in Table II (Section III), the minimal information set required for re-running workflows from a particular RO includes: i) a workflow file, and ii) input data for running the workflow. For the workflow runnability test, we needed to add two additional components to the RO:

1) **A workflow experiment description**, which uses the RO ontology to describe the processes and data used by the preserved workflow, which indicates samples of all the data required to run the workflow.
2) **A Minim description file in RDF**, to prescribe what constraints must be satisfied by the RO in order to enable workflow execution.

At the moment both files have to be manually created, but we envisage automating this operation. Table IV summarises the test result for our resulting ROs. 2 out of 4 passed our runnability test: all the ROs included the must-have workflow templates, but only 2 of them included the input data value

[8]http://www.w3.org/TR/turtle/

[9]http://www.wf4ever-project.org/wiki/display/docs/myExperiment+Pack+Analysis

TABLE III
SUMMARY OF THE PACKS USED FOR THE CASE STUDY.

Pack	Domain	No. of workflows	Example inputs	Example outputs	Example provenance traces
55	Genomics	3	n/a	Yes	n/a
58	Genomics	1	n/a	n/a	n/a
217	Data from the Time Series Data Library	2	As annotations in the workflow	As screenshots	n/a
219	Geography	3	No inputs were required	As screenshots	n/a

required for running the workflow. The assessment report of the ROs that failed the test (packs 55 and 58) clearly showed that their failure was due to the missing inputs.

However, passing the runnability test does not guarantee that the workflows can indeed be run. Our checklist defines the minimial requirements for a workflow to be runnable. In fact, when we tried to actually execute these workflows, none of them could be run straightforwardly using the information included in the packs/ROs, as summarised in Table IV.

TABLE IV
SUMMARY OF THE RUNNABILITY TEST RESULTS.

RO	Runnability test	Runnable	Decay cause
55	Failed, but fixed	No, only after we recovered the input data from the publication	Missing example data
58	Failed	No, services are no longer accessible	Inaccessible/Unavailable third-party resources & Insufficient workflow descriptions
217	Passed	No, but after we updated the scripts to process the updated data resources	Updates of third-party resources
219	Passed	Not completely, missing a local R library setting up	Missing execution environment

This shows that apart from the minimal set of information (workflow description and input data), individual workflows would really benefit from the preservation of additional information in order to deal with the specific type of decay, for example, the additional dependent libraries or sample output data.

B. Result replicability

The goal of our replicability test is to evaluate whether the RO contains sufficient information for assessing result replication has been achieved. Therefore, the minimal set of information that must be included in a test RO should include i) the information required for running an RO, as shown above, and ii) the outputs from a previous run. Therefore, in order to enable the replicability test, we need to add one more requirement to the Minim description files, which specifies that all the output data values of the preserved workflow must be bundled or referenced from within the RO.

Our replicable test results (summarised in Table V) showed that 3 of the ROs passed the replicability test. The RO for pack 58 is not runnable at all and contains neither input nor output data values. Of the ROs that passed the tests, we tested their actual replicability by running them. Our results showed that none of them managed to exactly produce the original results, as summarised in Table V.

TABLE V
SUMMARY OF THE REPLICABILITY TEST RESULTS.

RO	Replicability test	Replicability	Decay cause
55	Passed	No, not able to find the exact input data & produced different outputs	Missing example data & Insufficient workflow descriptions
58	Failed	Workflow is not runnable	Inaccessible/Unavailable third-party resources & Insufficient workflow descriptions
217	Passed	No, produced a more updated citation record	Updates of third-party resources
219	Passed	No, produced more updated weather forecast and not able to produce another part of the result due to missing R library	Updates of third-party resources

C. Summary

Our experiment shows that the minimal set of information is not sufficient for guaranteeing the actual runnability or replicability of our test workflows. An additional, customised set of information is needed, to support the specific needs, such as the requirement for the R library. However, our checklist tool did enable us to guarantee the completeness of an RO against an identified list of requirements and its report provided sufficient details for us to understand the causes of their failures. The size of our test sample, although small, allow us to assess the usefulness of our checklist. That said, creating a bigger corpus of ROs to evaluate our approach is a major part of our future work.

VIII. CONCLUSION

Decay is one of the main impediments to workflow reuse and to the reproducibility of workflow results. In this paper, we identified and characterised some causes of decay. Rather than speculating on the causes, we grounded our analysis upon an empirical study whereby we analysed real scientific workflows from the myExperiment repository. Our resulting decay classification allowed us to elaborate a concrete and pragmatic solution for mitigating workflow decay. Specifically, we proposed a framework that assists workflow publishers in identifying and bundling up the information necessary to prevent and help repairing workflow decay. As well as catering for minimal decay requirements, the framework that we propose provides means for assessing the quality of the bundle. Furthermore, such a framework is extensible; it allows designers to specify requirements that are particular to their workflows and the environment of their execution. Our approach is, to our knowledge, the first that provides a comprehensive understanding of workflow decay starting from the sources that causes the decay to a practical solution that enables decay to be addressed.

In our study, we have focused on Taverna workflows. However, we believe that our results are applicable to other scientific workflow systems, whose workflows are also composed of steps that rely on third party resources. To assess the degree to which this hypothesis holds in practice, we envisage, as part of our future work, to re-run our decay analysis using an expanded corpus of workflows, using Taverna workflows as well as workflows from other scientific workflow systems, such as Kepler [2], VisTrails [8] or Wings [17].

Acknowledgment

The research reported in this paper is supported by the EU Wf4Ever project (270129) funded under EU FP7 (ICT-2009.4.1).

References

[1] Open archives initiative object reuse and exchange, 2008.

[2] I. Altintas, O. Barney, and E. Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. *Provenance and annotation of data*, pages 118–132, 2006.

[3] S. Bechhofer, I. Buchan, D. De Roure, P. Missier, J. Ainsworth, J. Bhagat, P. Couch, D. Cruickshank, M. Delderfield, I. Dunlop, M. Gamble, D. Michaelides, S. Owen, D. Newman, S. Sufi, and C. Goble. Why linked data is not enough for scientists. *Future Generation Computer Systems*, 2011.

[4] Khalid Belhajjame. Semantic replaceability of escience web services. In *eScience*, pages 449–456. IEEE Computer Society, 2007.

[5] Khalid Belhajjame, Oscar Corcho, Daniel Garijo, Jun Zhao, Paolo Missier, David Newman, Raul Palma, Sean Bechhofer, Esteban Garc Cuesta, Jose Manuel Gomez-Perez, Graham Klyne, Kevin Page, Marco Roos, Jose Enrique Ruiz, Stian Soiland-Reyes, Lourdes Verdes-Montenegro, David De Roure, and Carole A. Goble. Workflow-centric research objects: First class citizens in scholarly discourse. In *Proceeding of SePublica2012*, pages 1–12, 2012.

[6] Khalid Belhajjame, Carole A. Goble, Stian Soiland-Reyes, and David De Roure. Fostering scientific workflow preservation through discovery of substitute services. In *eScience*, pages 97–104. IEEE Computer Society, 2011.

[7] P.E. Bourne, T. Clark, R. Dale, A. De Waard, I. Herman, E. Hovy, D. Shotton, et al. Improving future research communication and e-scholarship: a summary of findings— macquarie university researchonline. 2012. http://force11.org/white_paper.

[8] S.P. Callahan, J. Freire, E. Santos, C.E. Scheidegger, C.T. Silva, and H.T. Vo. Vistrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747. ACM, 2006.

[9] F. Calore, D. Lombardi, Enrico Mussi, Pierluigi Plebani, and Barbara Pernici. Retrieving substitute services using semantic annotations: A foodshop case study. In *Business Process Management Workshops*. Springer, 2007.

[10] Z. Chen and L. Moreau. Implementation and evaluation of a protocol for recording process documentation in the presence of failures. *Provenance and Annotation of Data and Processes*, pages 92–105, 2008.

[11] S. Chin, K. Kim, and Y.S. Kim. A process-based quality management information system. *Automation in Construction*, 13(2):241–259, 2004.

[12] 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.

[13] D. Crawl and I. Altintas. A provenance-based fault tolerance mechanism for scientific workflows. *Provenance and Annotation of Data and Processes*, pages 152–159, 2008.

[14] D. De Roure, C. Goble, and R. Stevens. The design and realisation of the myexperiment virtual research environment for social sharing of workflows. *Future Generation Computer Systems*, 25:561–567, 2009.

[15] M. Gamble, C. Goble, G. Klyne, and J. Zhao. Mim: A minimum information model vocabulary and framework for scientific linked data. 2012. in submission.

[16] Yolanda Gil, Ewa Deelman, Mark H. Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole A. Goble, Miron Livny, Luc Moreau, and Jim Myers. Examining the challenges of scientific workflows. *IEEE Computer*, 40(12):24–32, 2007.

[17] Yolanda Gil, Varun Ratnakar, Jihie Kim, Pedro Antonio Gonzalez-Calero, Paul Groth, Joshua Moody, and Ewa Deelman. Wings: Intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems*, 26(1), 2011.

[18] F. Giunchiglia and R. ChenuAbente. Scientific knowledge objects v. 1. Technical report, Technical Report DISI-09-006, University of Trento, 2009.

[19] Carole A. Goble, David De Roure, and Sean Bechhofer. Accelerating scientists' knowledge turns. In *Proceedings of The 3rd international IC3K joint conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management.*, 2012. in press.

[20] P. Groth, A. Gibson, and J. Velterop. The anatomy of a nanopublication. *Information Services and Use*, 30(1):51–56, 2010.

[21] B.M. Hales and P.J. Pronovost. The checklist–a tool for error management and performance improvement. *Journal of critical care*, 21(3):231–235, 2006.

[22] J. Hunter. Scientific publication packages–a selective approach to the communication and archival of scientific output. *International Journal of Digital Curation*, 1(1):33–52, 2008.

[23] S. Köhler, S. Riddle, D. Zinn, T. McPhillips, and B. Ludäscher. Improving workflow fault tolerance through provenance-based recovery. In *SSDBM*, pages 207–224. Springer, 2011.

[24] D. Koop, E. Santos, P. Mates, H.T. Vo, P. Bonnet, B. Bauer, B. Surer, M. Troyer, D.N. Williams, J.E. Tohline, et al. A provenance-based infrastructure to support the life cycle of executable papers. *Procedia CS*, 4:648–657, 2011.

[25] Kevin Lee, Rizos Sakellariou, Norman W. Paton, and Alvaro A. A. Fernandes. Workflow adaptation as an autonomic computing problem. In *Proceedings of the 2nd workshop on Workflows in support of large-scale science*, WORKS '07, pages 29–34, New York, NY, USA, 2007. ACM.

[26] R. Littauer, K. Ram, B. Ludäscher, W. Michener, and R. Koskela. Trends in use of scientific workflows: Insights from a public repository and recommendations for best practices. In *The Seventh International Digital Curation Conference*, 2011.

[27] Phillip Mates, Emanuele Santos, Juliana Freire, and Cláudio T. Silva. Crowdlabs: Social analysis and visualization for the sciences. In *SSDBM*, pages 555–564. Springer, 2011.

[28] B. Matthews, A. Shaon, J. Bicarregui, and C. Jones. A framework for software preservation. *International Journal of Digital Curation*, 5(1), 2010.

[29] Paolo Missier, Stian Soiland-Reyes, Stuart Owen, Wei Tan, Aleksandra Nenadic, Ian Dunlop, Alan Williams, Tom Oinn, and Carole A. Goble. Taverna, reloaded. In *SSDBM*, pages 471–481. Springer, 2010.

[30] David Newman, Sean bechhofer, and David De Roure. myexperiment: An ontology for e-research. In *Workshop on Semantic Web Applications in Scientific Discourse in conjunction with the International Semantic Web Conference*, 2009.

[31] Thomas M. Oinn and Et Al. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.

[32] David De Roure, Khalid Belhajjame, and Et Al. Towards the preservation of scientific workflows. In *Procs. of the 8th International Conference on Preservation of Digital Objects (iPRES 2011)*. ACM, 2011.

[33] C.F. Taylor, D. Field, S.A. Sansone, J. Aerts, R. Apweiler, M. Ashburner, C.A. Ball, P.A. Binz, M. Bogue, T. Booth, et al. Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the mibbi project. *Nature biotechnology*, 26(8):889–896, 2008.

[34] P. Van Gorpa and S. Mazanekb. Share: a web portal for creating and sharing executable research papers. *Procedia Computer Science*, 4:589–597, 2011.

[35] I. Wassink, P.E. van der Vet, K. Wolstencroft, P.B.T. Neerincx, M. Roos, H. Rauwerda, and T.M. Breit. Analysing scientific workflows: Why workflows not only connect web services. In *2009 World Conference on Services - I*, pages 314 –321, 2009.

IX. Appendix

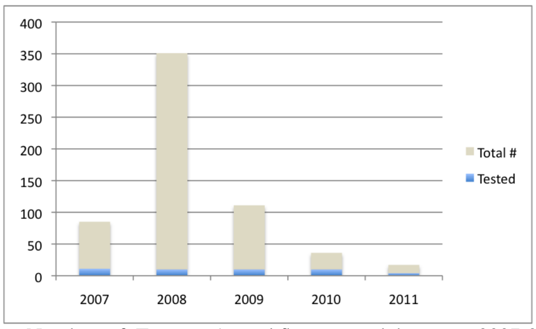

Fig. 3. Number of Taverna 1 workflows tested between 2007-2011, in comparison to total number of workflows from that year in myExperiment.

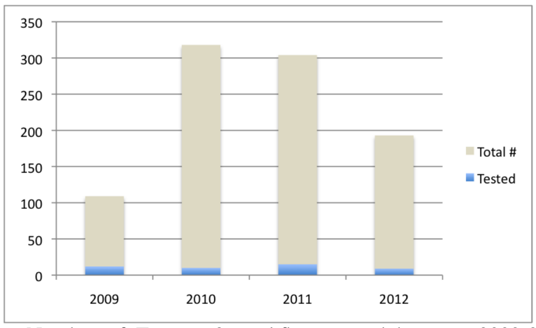

Fig. 4. Number of Taverna 2 workflows tested between 2009-2012, in comparison to total number of workflows from that year in myExperiment.

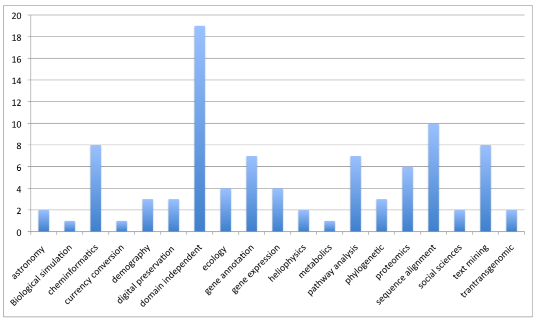

Fig. 5. Distribution of number of workflows from each different domain.

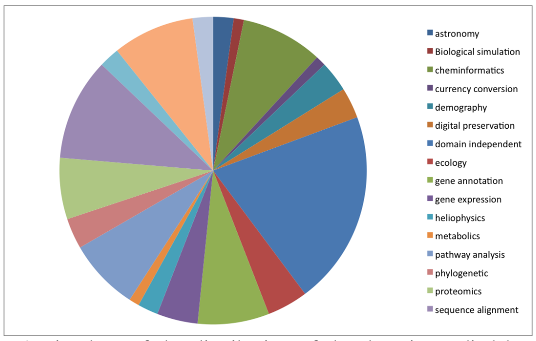

Fig. 6. A pie chart of the distribution of the domain studied by our test workflows.

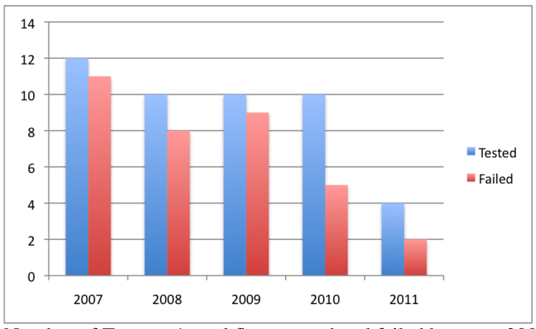

Fig. 7. Number of Taverna 1 workflows tested and failed between 2007-2009.

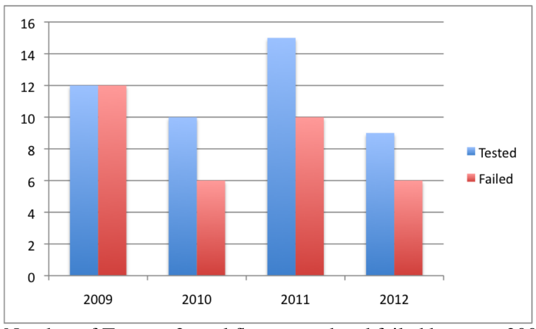

Fig. 8. Number of Taverna 2 workflows tested and failed between 2009-2012.

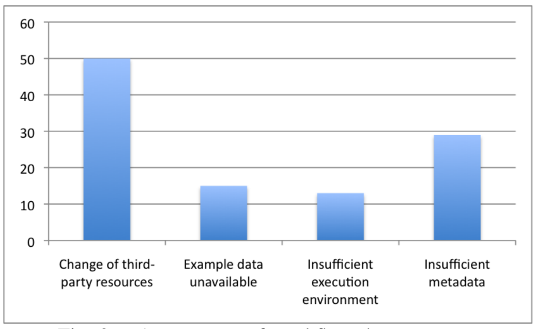

Fig. 9. A summary of workflow decay causes.

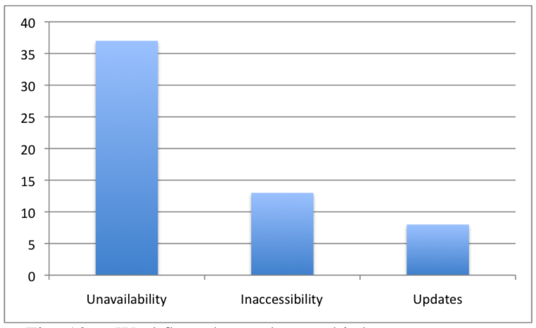

Fig. 10. Workflow decay due to third party resources.