Towards FAIR protocols and workflows: The OpenPREDICT use case

Remzi Celebi[1,*], **Joao Rebelo Moreira**[2,*], **Ahmed A. Hassan**[3], **Sandeep Ayyar**[4], **Lars Ridder**[5], **Tobias Kuhn**[2], and **Michel Dumontier**[1]

[1]**Institute of Data Science, Maastricht University, Maastricht, Netherlands**
[2]**Computer Science, VU University Amsterdam, Amsterdam, Netherlands**
[3]**Pharmacology & Personalised Medicine, Maastricht University, Maastricht, Netherlands**
[4]**Medical Informatics, Stanford University, Palo Alto, California, USA**
[5]**Netherlands eScience Center, Amsterdam, Netherlands**

Corresponding author:
Remzi Celebi, Joao Rebelo Moreira[1,2]

Email address: remzi.celebi@maastrichtuniversity.nl, j.luizrebelomoreira@utwente.nl

ABSTRACT

It is essential for the advancement of science that researchers share, reuse and reproduce each others workflows and protocols. The FAIR principles are a set of guidelines that aim to maximize the value and usefulness of research data, and emphasize a number of important points regarding the means by which digital objects are found and reused by others. The question of how to apply these principles not just to data but also to the workflows and protocols that consume and produce them is still under debate and poses a number of challenges. In this paper we describe an inclusive and overarching approach to apply the FAIR principles to workflows and protocols and demonstrate its benefits. We apply and evaluate our approach to make the PREDICT workflow, a highly cited drug repurposing workflow, open and FAIR. This includes FAIRification of the involved datasets, as well as applying semantic technologies to represent and store data about the detailed versions of the general protocol, of the concrete workflow instructions, and of their execution traces. A semantic model was proposed to better address these specific requirements and was evaluated by answering competency questions. This semantic model consists of classes and relations from a number of existing ontologies, including Workflow4ever, PROV, EDAM, and BPMN. This allowed us then to formulate and answer new kinds of competency questions. Our evaluation shows the high degree to which our FAIRified OpenPREDICT workflow now adheres to the FAIR principles and the practicality and usefulness of being able to answer our new competency questions.

1 INTRODUCTION

Reproducible results are one of the main goals of science. A recent survey, however, showed that more than 70% of researchers have been unsuccessful in trying to reproduce another research experiment and more than 50% failed to reproduce their own research studies (Baker, 2016). Failure to represent reproducible methodology is a major detriment to research.

The rate of non-reproducibility for pharmacological studies is particularly worrying. Together with the incredible expense (over $1 billion US) and high rate of failure (90%), it highlights the need for new approaches in drug discovery (Scannell et al., 2012). Therefore, pharmacology makes a very good example and use case for the work we present here. Specifically, we will be looking into drug repositioning, where small molecules approved for one indication are repurposed for a new indication. Drug repositioning is gaining recognition as a safe, effective, and much lower-cost approach to uncover new drug uses (Ashburn and Thor, 2004; Sleigh and Barton, 2010). The availability of public data, both in the form of literature curated knowledge and 'omics data has created exciting opportunities for computational drug repositioning. For instance, gene expression data in repositories such as Gene Expression Omnibus (GEO) enable the analysis of correlation between drug and gene expression – termed the Connectivity

47 Map approach - to find chemicals that may counter cellular disorders (Barrett and Edgar, 2006), including
48 Alzheimer's, and small cell lung cancer (Lamb et al., 2006; Sirota et al., 2011). More sophisticated
49 approaches use network analysis and machine learning to efficiently combine drug and disease data
50 (Cheng et al., 2012; Gottlieb et al., 2011; Hoehndorf et al., 2013; Wu et al., 2013; Bisgin et al., 2014).

51 The ability to reproduce original research results is contingent on the availability of the original data,
52 methods and results to the research community. The FAIR principles (Wilkinson et al., 2016), describe a
53 set of requirements for data management and stewardship to make research data Findable, Accessible,
54 Interoperable, and Reusable. Ongoing efforts on FAIR cover data policies, data management plans,
55 identifier mechanisms, standards and data repositories (Collins et al., 2018). Highly diverse communities,
56 from the biomedical sciences to the social sciences and humanities, are now working towards defining
57 standards for publication and sharing of data. In anticipation, new methods and infrastructure are needed
58 to facilitate the generation of FAIR data and workflows.

59 Here, we describe a methodology to publish scientific workflows as FAIR data, whether they involve
60 manual or computational steps. We evaluate our method by applying it to the PREDICT drug repositioning
61 workflow. Based on this example, we will try to answer our research question of how we can use existing
62 vocabularies and techniques to make scientific workflows more open and FAIR, with a particular focus
63 on the interoperability aspect. The main contributions of this paper are (a) general guidelines to make
64 scientific workflows open and FAIR, focusing on the interoperability aspect, (b) the OpenPREDICT use
65 case, demonstrating the open and FAIR version of the PREDICT workflow, (c) new competency questions
66 for previously unaddressed reproducibility requirements, and (d) evaluation results on the practicality and
67 usefulness of our approach.

68 ## 2 BACKGROUND

69 Below, we refer to the most relevant background with respect to reproducibility, workflow systems, and
70 applying FAIR to workflows.

71 ### 2.1 Reproducibility of Scientific Workflows

72 Reproducibility is a persistent issue in many scientific studies. A recent study showed that data scientists
73 spend 19% of their time finding, understanding and accessing datasets, and 60% of their time cleaning
74 and organizing these datasets to use in their studies (CrowdFlower, 2016). Thus, only 20% of the time is
75 left for data scientists to spend on their core activities, such as mining data, refining algorithms, building
76 training sets and analyzing the results. To address these data sharing problems (Poldrack and Poline,
77 2015; Ioannidis, 2005b), systematic reviews and meta-analyses have recently received increased attention
78 to solve some of these problems (Moher et al., 2009) and several initiatives emphasize the methods to
79 share and reuse these workflows.

80 According to the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) (Borgo
81 and Masolo, 2010), a workflow is a "plan that defines role(s), task(s), and a specific structure for tasks to
82 be executed, usually supporting the work of an organization", and a plan is a description of instructions
83 with explicit goal(s). The lack of specific details of scientific workflows, including the specification of
84 instructions and their derived code with different steps and parameters, are factors believed to contribute to
85 a non-reproducibility rate of 64% for pharmacological studies (Ioannidis, 2005a; Prinz et al., 2011), 89%
86 in cancer (Begley and Ellis, 2012), and in 66% psychology (Klein et al., 2014). Usually these workflows
87 are often imprecisely detailed in scientific publications and therefore cannot be reliably reproduced
88 (Vasilevsky et al., 2013). A recent research over 1.4 million Jupyter notebooks (available in GitHub)
89 found that only 24.11% of the notebooks could be executed without exceptions and only 4.03% produced
90 the same results (Pimentel et al., 2019).

91 ### 2.2 Workflow Systems

92 To avoid the workflow decay phenomenon (Hettne et al., 2012), a number of recent initiatives target the
93 improvement of workflow reproducibility, for example initiatives under the Common Workflow Language
94 (CWL)[1], which has become the de facto standard for syntactic interoperability of workflow management
95 systems. CWL can be considered a general purpose language for workflow definition, dealing with
96 representation issues on data and control flows for a long time. Others have added semantic models and

---

[1]https://www.commonwl.org/

methods, for example the Workflow4ever project with its Research Objects method (Belhajjame et al., 2015), to improve interoperability and connect workflows to real-world entities in a systematic way.

These connections in a broad sense can be called the provenance of a workflow, which can be classified into prospective provenance, retrospective provenance, and workflow evolution provenance. Retrospective provenance refers to the information about executions, including the atomic events that consume inputs and produce outputs, as well as information about the execution environment (Khan et al., 2019). Prospective provenance refers to the specifications or "recipes" that describe the workflow steps and their execution order, typically as an abstract representation of these steps (protocols), as well as expected inputs and outputs (Cohen-Boulakia et al., 2017). Workflow evolution provenance refers to tracking the versions of workflows and the respective data, as the workflow is changed and improved over time.

A number of models and methods have been developed to capture these different kinds of provenance in the context of workflows. The PROV ontology (Lebo et al., 2013) provides the vocabulary and model for provenance in general, which can be used in conjunction with top-level ontologies such as DOLCE (Borgo and Masolo, 2010) and wide-spread versatile vocabularies such Dublin Core and schema.org. A number of approaches build upon PROV to apply it to workflows, such as the Open Provenance Model for Workflows (OPMW) (Moreau et al., 2008), P-PLAN (Garijo and Gil, 2012), and CWLProv (Soiland-Reyes et al., 2018). Other notable examples include ProvBook and the Reproduce-me ontology (Samuel and König-Ries, 2018a,b) to run workflows in Jupyter notebooks, the ML-Schema ontology for machine learning workflows (Correa Publio et al., 2018), SPAR Publishing Workflow Ontology (PWO) for workflows in scientific publications (Hartanto et al., 2017), and Business Process Modelling Notation (BPMN) ontology to specify business processes (Rospocher et al., 2014). We also highlight the SMART protocols initiative (Giraldo, 2017) targeting the description of laboratory protocols, such as SOP, and the protocols.io web-system, which allows describing and storing the execution traces.

2.3 Applying FAIR to Workflows

The FAIR principles have received significant attention, but little is known about how scientific protocols and workflows can be aligned with these principles. Making a workflow FAIR-compliant allows for general-purpose software to interpret and use the data. The application of FAIR in healthcare, for example, has shown that these principles can provide a boost to data-driven applications that require the integration of data coming from different sources, achieving "interoperability without the need to all speak exactly the same language" (Imming et al., 2018). To address the current lack of methods to make entire software workflows FAIR-compliant, some recent initiatives have outlined how FAIR can be applied to software (Neil and Daniel S., 2018; Lamprecht et al., 2019). The ultimate goal is to not just make input and output data FAIR, but the entire process in between, in order to fully address the challenges of sharing and reproducing results. This includes the precise description of all steps and parameters, which currently even human experts often can't reconstruct from what is published in scientific articles (Vasilevsky et al., 2013).

The FAIRification process Jacobsen et al. (2019) specifies the steps required to transform an existing data element to FAIR data, leveraged by the RDF technology. RDF is a broadly applicable formal language to achieve semantic interoperability, able to address principle I1. The FAIRification starts by retrieving the non-FAIR data from the source(s). Then, these datasets should be analyzed to enable the understanding of the data structures and how they are mapped to concepts from the domain. The next step, semantic modelling, is a major activity comprising semantic harmonisation and integration, requiring either the creation of new predicates or the reuse of semantic models adherent to the FAIR principles. Once the dataset is represented with semantic elements, the dataset is transformed to RDF to make data linkable, followed by the addition of licensing information and metadata definition (e.g., for indexed search). The last step is to store the FAIRified data into a FAIR Data Point, which can be implemented with a triplestore (e.g., Virtuoso and GraphDB).

3 THE FAIR WORKFLOWS APPROACH

In this section we describe the workflow representation requirements, giving focus on the novel parts of our overarching model, such as covering manual steps, different workflow abstraction levels (e.g., protocols such as standard operating procedures) and versioning on all these levels. We formulate these

149 requirements as competency questions and present a configuration of elements from existing semantic
150 models as a unified model to answer these competency questions.

3.1 Requirements: competency questions

152 With the help of structured interviews with data scientists and a gap analysis of the literature, we
153 formulated user requirements for the reproducibility of workflows (the details of the interviews were
154 given in Appendix 1). Those user requirements were essential to infer the design and implementation of
155 an ontology, particularly as competency questions that it should be able to answer.

156   The interviewees stated that they experience many challenges in reproducing their or others' work,
157 due to the lack of details of workflow steps, data cleaning and filtering. In addition to this, essential
158 information, such as processing parameters or design details needed to reproduce the results, is often
159 missing. Some of these requirements are already addressed by the existing approaches, others represent
160 gaps in the literature. The interviews showed that the definition of manual processes of workflows are
161 usually missing or incomplete, which is a requirement poorly addressed by computational workflow
162 approaches. Often, software libraries, packages and versions of tools used are not explicitly recorded. The
163 suggestions given by the data scientists for a generic way of defining computational and manual workflows
164 were to make metadata of the datasets accessible, add richer prospective and retrospective provenance,
165 and allow for fine-grained workflow versioning linked to outputs produced during distinct executions. A
166 unanimous recommendation is to allow for the separate input of relevant workflow parameters, so that
167 one can run the same workflow multiple times to apply different processing options without having to
168 change the workflow itself.

169   The representation of software environment details (e.g., libraries and packages) is already addressed
170 by some of the surveyed semantic models, like Workflow4ever, CWLProv and Reproduce-me. We revised
171 all capabilities of the existing semantic approaches to address the needs collected from the interviews.
172 Therefore, we looked for the gaps in these semantic models and checked whether critical issues could
173 impede their reuse (e.g., ontology consistency checks). From this study, we conclude that none of the
174 related work could completely address all the requirements together. We clustered the gaps in three
175 main categories: (CQ1) Manual steps description and executions; (CQ2) abstraction levels of workflows;
176 and (CQ3) versioning of executed workflows. Therefore, we propose the following additional sets of
177 competency questions (CQ):

178 CQ1 Questions about manual steps:

179   CQ1.1 Which steps are meant to be executed manually and which to be executed computationally?

180   CQ1.2 For the manual steps, who are the agents responsible to execute them (individuals and roles)?

181   CQ1.3 Which datasets were manually handled and their respective formats?

182   CQ1.4 What are the types of manual steps involved?

183 CQ2 Questions about instantiation of general workflows by more specific ones:

184   CQ2.1 What are the main steps of a general workflow?

185   CQ2.2 What are the steps of a specific workflow and how are they described?

186   CQ2.3 What higher-level description instantiated a certain workflow step?

187   CQ2.4 Who or what method made the instantiation of a semantic/meta level description of a step
188   into an executable workflow step?

189 CQ3 Questions about versioning of workflows and their executions:

190   CQ3.1 What are the existing versions of a workflow and what are their provenance?

191   CQ3.2 Which instructions were removed/changed/added from one version to another?

192   CQ3.3 Which steps were automatized from one version to another?

193   CQ3.4 Which datasets were removed/changed/added for the different versions?

194   CQ3.5 Which workflow version was used in each execution and what was generated?

195   To the best of our knowledge, none of the previous research on semantic modelling of workflows (or
196 protocols/processes) address all these requirements together, in few cases some semantic models only
197 partially cover some questions, as explained in the prior section.

3.2 Unified Model

199 From the study of the diverse existing semantic models for workflows and protocols, we compiled a
200 unified conceptual model covering the elements required to answer our competency questions. For this,
201 we applied the ontology-driven conceptual modelling approach (Guizzardi et al., 2015), which is based on
202 the Unified Foundational Ontology (UFO) and its ontological language OntoUML (Moreira et al., 2016).
203   Figure 1 illustrates the main elements of our unified model[2], which is mainly based on DOLCE
204 Ultra Lite (DUL), PROV, P-PLAN and BPMN 2.0. The most relevant ontology used is P-PLAN, which
205 provides an abstract terminology of the main building blocks to describe plans.
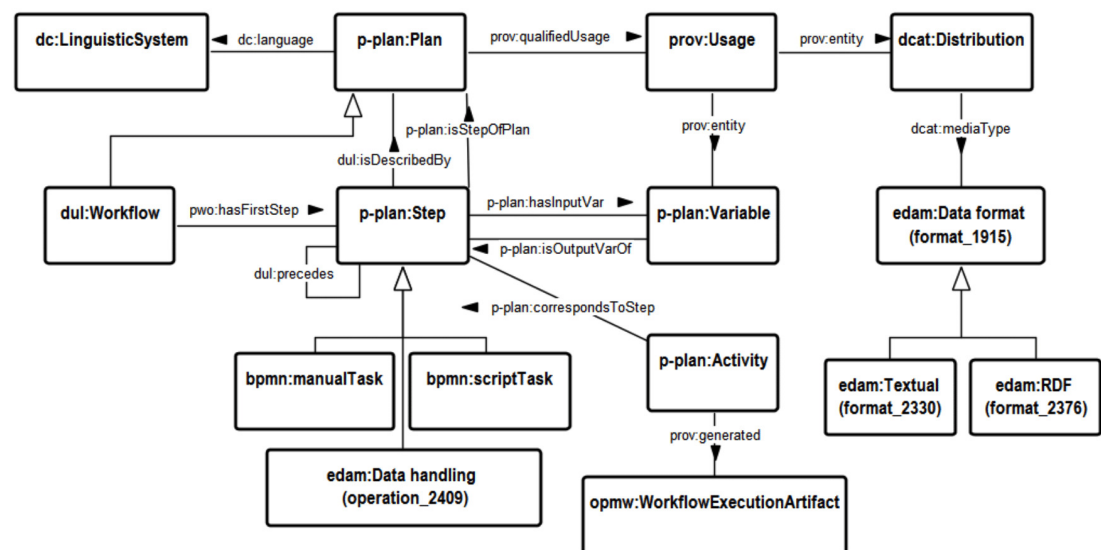
Figure 1. Unified semantic model for workflows

206   The *p-plan:Plan* category is the core element of our unified model and is the class used to classify any
207 type of instruction. It allows for the composition of instruction by means of smaller steps (*p-plan:Step*)
208 that have input and output variables (*p-plan:Variable*). With the *pwo:hasFirstStep* property, we can
209 indicate the first step of a plan, and with *dul:precedes*, we can indicate whenever a step precedes another,
210 thereby enabling the representation of sequential and parallel steps.
211   We decouple a particular step within a workflow from its instruction with the pattern *p-plan:Step*
212 *dul:isDescribedBy p-plan:Plan*, where each step always point to one plan. This approach allows us to
213 separate the workflow steps, enabling the reuse of instructions by different workflows. Therefore, in
214 our approach a step is a lightweight object (like a pointer) that serves only for ordering of instructions
215 without coupling them to the specific workflow. Besides that, we use the *dul:isDescribedBy* property
216 as a self-relationship of *p-plan:Plan*, to represent that an instruction describes another instruction in
217 a different abstraction level. With this approach, we can represent anything from high-level abstract
218 protocols to concrete and executable workflow steps, and the links between these levels. This can be
219 used to first represent the general protocol and then move to the definition of the executable steps akin to
220 the common software engineering phases of specification and implementation. Our model can however
221 also be used in the other direction to extract a new common protocol from similar existing concrete

---

[2]https://w3id.org/fair/plex

workflows. At the more abstract levels, instructions are written in a natural language like English (or possibly pseudo-code), whereas at the lowest level, we find the executable specifications, which can be written in a programming language and thereby automatically executable. Also at the lowest level, however, instructions can be in natural language, such as for wet-lab instructions, which can naturally only be executed in a manual fashion. For example, the first general step of a specification of a machine learning pipeline like OpenPREDICT (to which we will come back to shortly) might be to "load all features and gold standard" (a *p-plan:Plan*). The concrete execution of this general step is described by four concrete and executable steps (written in a language such as Python), each having a link (*dul:isDescribedBy*) to the general description of the step.

We use the BPMN 2.0 ontology for the representation of manual and computational activities with *bpmn:ManualTask* and *bpmn:ScriptTask*, which we both define as subclasses of *p-plan:Step*. With this approach, the modeller can therefore include manual and automated steps in the same workflow. More specific classes can be used for particular workflow systems, such as *reprod:Cell* as a kind of *bpmn:ScriptTask* describing a code cell in a Jupyter Notebook.

We follow the FAIR Data Point specification[3] for the representation of datasets (input and output) through the *dcat:Dataset* element, which should be linked to the available distributions (*dcat:Distribution*) through the *dcat:distribution* property, and the URL to download the distribution is represented with *dcat:downloadURL*. We improved this approach with data formats from the EDAM ontology through the *dcat:mediaType* property. We use *prov:qualifiedUsage* for variable bindings. For example, the instruction (*p-plan:Plan*) to "download a dataset and save it in the local environment" has a link (*prov:qualifiedUsage*) to the "binding the online dataset to a local variable" (*prov:Usage*), which represents the connection between the dataset distribution (*dcat:Distribution*) and the local variable (*p-plan:Variable*) through instances of the *prov:entity* properties. For the representation of retrospective provenance, i.e., information about prior executions of a workflow, we follow the P-PLAN approach by using *p-plan:Activity* and linking it to the steps with *p-plan:correspondsToStep*.

To represent the roles of the different involved agents (such as people and software), we use the agent associations as defined in PROV. For example, the Jupyter Notebook (*prov:SoftwareAgent*) was used as execution environment (*prov:Role*) for all computational steps of the OpenPREDICT workflow.

Furthermore, as a practical design decision, we extended the notion of *prov:Association* for endurants, so the modeller can apply the association pattern similarly to the perdurant way, i.e., use the property *prov:hadPlan* from *p-plan:Association* to *p-plan:Plan* instead of the relation from *prov:Activity* through *prov:qualifiedAssociation*. Therefore, this approach allows the modeller to represent the association of agent roles to an instruction. For example, Remzi is the OpenPREDICT main developer, so the "Remzi as developer of OpenPREDICT" (*prov:Association*) links to (a) the "Developer" (*prov:Role*) through *prov:hadRole* property, (b) the Remzi object (a *prov:Agent*) through *prov:agent*; and (c) all OpePREDICT instructions (*p-plan:Plan*), through *prov:hadPlan*. Notice that, although the terminology of these properties targeted the perdurant aspect (*prov:Activity*), these properties are also useful for the endurant aspect. Ideally, they should have the adequate endurant terminology, so instead of prov:hadPlan, it should be "*prov:hasPlan*" (similarly for *prov:hadRole* too).

One of the most important links is of course the one between a workflow execution and its created outputs. For this, we specialized the PROV approach by using *prov:generated* to link a workflow activity (*p-plan:Activity*) to an output artifact (*opmw:WorkflowExecutionArtifact*). Therefore, each step execution can generate workflow execution artifacts. To represent the specifics of machine learning workflows, we moreover use the ML-Schema ontology (mls), such as to specify the trained model and its evaluation measures (via *mls:ModelEvaluation* and *mls:EvaluationMeasure*). For example, it can be used to specify the accuracy of the models trained during different executions.

For the representation of versioning, finally, we use *dc:hasVersion* to assign version identifiers and *prov:wasRevisionOf* to link to the previous versions, and apply this to all relevant elements, including workflows, instructions, software, and datasets.

3.3 Case Study Topic

We evaluate our approach below with a case study of a computational drug repurposing method based on machine learning, called PREDICT (Gottlieb et al., 2011). PREDICT is one of the most frequently cited drug repurposing methods and provides a ranking of drug disease associations on the basis of their

---

[3]https://github.com/FAIRDataTeam/FAIRDataPoint-Spec

similarity to a set of known associations. PREDICT has reported a high AUC (0.90) for predicting drug indications, however neither the original data nor the software to produce the results are available to directly replicate this computational study.

The features for the drug prediction classifier included of five drug–drug similarity measures and two disease–disease similarity measures. The similarities between drugs were calculated based on molecular fingerprints, common side effects of drugs, target protein sequence alignment, semantic similarity of target genes of drugs in the Gene Ontology, and closeness of target proteins in human protein–protein interaction network. For the disease aspect, two disease–disease similarities were calculated based on medical description of diseases and semantic similarity of disease terms in the Human Phenotype Ontology. The method transforms drug–drug and disease–disease similarities into integrated features to be used for a logistic regression training.

For evaluating the performance of the logistic regression, 10-fold cross-validation was used in two different ways: One in which 10% of drugs are hidden and one in which 10% of associations are hidden. In the first strategy, 10% randomly selected drugs in the gold standard and the known indications associated with them were removed. The positive training set consisted of the remaining 90% of drugs and the indications associated with them. The negative training set consisted of randomly generated drug-disease associations which were not in the positive set. For the second strategy, the known associations were divided into 90% positive training and 10% positive test sets, while negative training and test sets were built using randomly generated drug-disease associations from respective sets.

In the next section, we show report on the application of our approach on this use case.

4 OPENPREDICT CASE STUDY

As case study, we took the original PREDICT workflow, as introduced above, and transformed it with our approach to make it open and FAIR. We therefore call the resulting workflow OpenPREDICT. It implements the same steps of the original PREDICT, i.e., five drug–drug similarity and two disease–disease similarity measures were used to train a logistic classifier to predict potential drug-disease association (see Figure 2). Therefore, we follow the same general protocol of these four steps:

1. **Data preparation:** In this step, necessary data set is collected and preprocessed.

2. **Feature Generation:** In this step, we generate features from the collected data sets. Drug-drug and disease-disease similarity scores were combined by computing the weighted geometric mean. Thus, we combine 5 drug-drug similarity measures and 2 disease-disease similarity measures, resulting in 10 features.

3. **Model Training:** In this step, the generated features from previous step are to be used to train in a simple logistic classifier.

4. **Model Evaluation:** This step uses 2 different cross-validation approaches: One where 10 % of drugs is hidden and one where 10 % of associations is hidden for test. ROC AUC, AUPR, accuracy, precision and F-score of the classifier on test data are reported.

Below we explain how we made a FAIR version of PREDICT's input data and then show how we used our approach to model the OpenPREDICT workflow that is consuming this data. The implementation and the workflow description of OpenPREDICT are available at GitHub [4].

4.1 FAIRified Data Collection

Since the original data used in PREDICT is not publicly available, we collected data from open sources and made it FAIR with Linked Data (Bizer et al., 2009) representations. We obtained data about drugs, drug targets, drug chemical structure, and drug target sequence from DrugBank (Wishart et al., 2008), and additional drug targets from the KEGG dataset (Kanehisa et al., 2007). The SIDER dataset (Kuhn et al., 2010) was used for drug side effects and the HGNC and the GOA datasets (Gray et al., 2015; Barrell et al., 2009) were used for gene identifier mapping and gene ontology (GO) annotation respectively. We used Linked Data versions of the above-mentioned datasets from Bio2RDF (Callahan et al., 2013), which is an influential resource for the biomedical sciences, providing a network of data collected form several major

---

[4]https://github.com/fair-workflows/openpredict
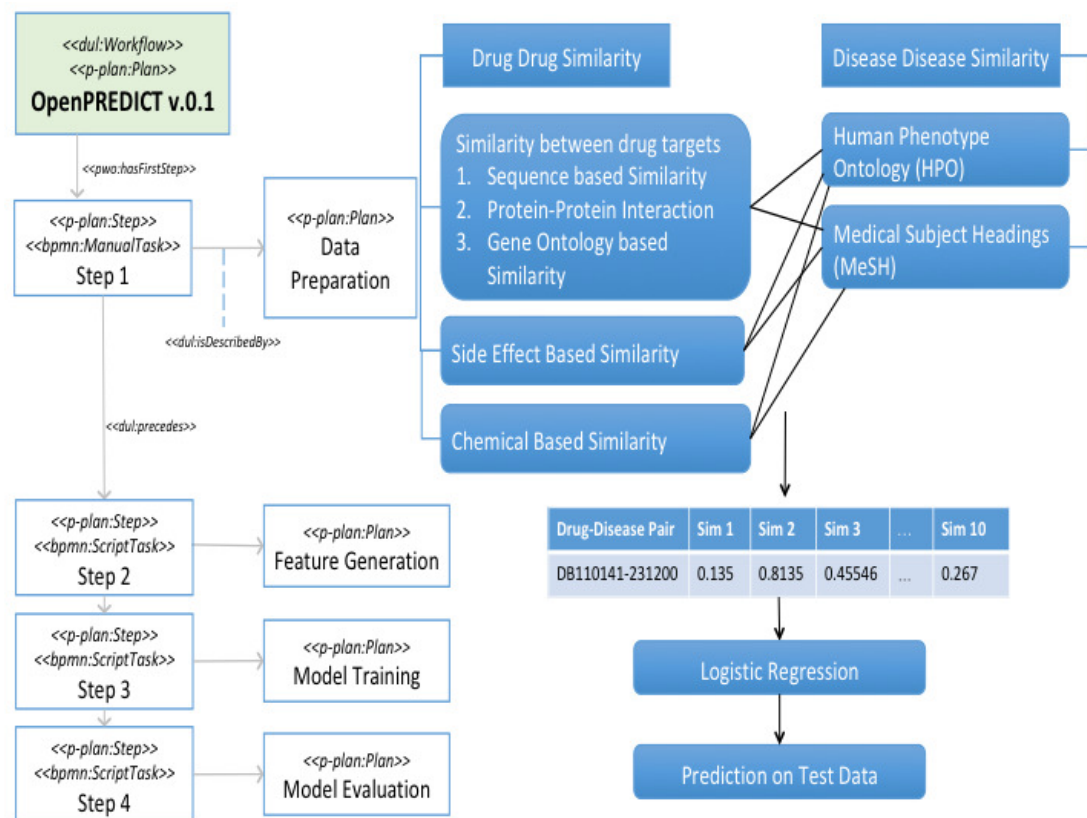
Figure 2. OpenPREDICT Workflow (version 0.1) with manual and computational steps .

biological databases. On to of that, we used the supplementary file provided by (Menche et al., 2015) for protein–protein interactions and disease phenotype annotations that link HPO terms to OMIM diseases[5]. MeSH annotations were collected from (Caniza et al., 2015)[6] and annotations were also obtained by NCBO annotator API (Noy et al., 2009) using the OMIM disease description.

The data that were not yet in a Linked Data format were converted to RDF with a FAIRification process (Jacobsen et al., 2019). We kept the copies of the retrieved non-RDF datasets in our GitHub repository to prevent the data access issues that may arise due to the unavailability of the data sources. We also stored the collected datasets in a triplestore and created SPARQL queries to access the triplestore in order to produce the features for PREDICT's method.

Our OpenPREDICT workflow has two versions (0.1 and 0.2). In the first, we experimented with the FAIRifier tool with the two inputs that are provided as text files, i.e., (protein–protein interactions in human interactome) and (disease phenotypic descriptions). Besides the formalization of the manual steps through our approach, we also provide guidelines for the manual steps. In the second, we wrote Python scripts for FAIRificiation process of these datasets, evolving most of the manual steps to computational ones. Table 1 summarizes the list of all datasets used in version v0.1 and v0.2. OpenPREDICT v0.2 also uses a different MESH annotation dataset for disease similarity (indicated with † in Table 1)

For this FAIRification process, the data needs to be mapped to formal semantic models. In our case, important concepts included "protein-protein interaction" from Bioportal, "protein interactions" from EDAM (*edam:topic_0128*), the Bio2RDF properties *bio2rdf:interactor_a bio2rdf:interactor_b* for the gene interactors representing the role of a gene in a protein–protein interaction, and "disease" and "has phenotype" from SIO (SIO_010299 and SIO_001279).

---

[5]https://hpo.jax.org/app/download/annotation
[6]https://paccanarolab.org/disease_similarity

Dataset file	Date Retrieved	Data format	Download URL
Bio2RDF r4 datasets (Drugbank, KEGG, HGNC, SIDER and GOA)	2019-08-15	.nq (RDF) compressed as .gz	`https://download.bio2rdf.org/#/release/4/`
PREDICT drug indication gold standard	2019-08-15	.tab with tabular separator	`https://www.ncbi.nlm.nih.gov/pmc/article/PMC3159979/bin/msb201126-s4.xls`
Pubchem-Drugbank mappings	2019-08-15	.tab with tabular separator	`https://raw.githubusercontent.com/dhimmel/drugbank/gh-pages/data/mapping/pubchem.tsv`
Protein-protein interactions	2019-08-15	.txt with tabular separator	`https://media.nature.com/full/nature-assets/srep/2016/161017/srep35241/extref/srep35241-s3.txt`
HPO Phenotype annotations	2019-08-15	.tab with tabular separator	`http://compbio.charite.de/jenkins/job/hpo.annotations/lastSuccessfulBuild/artifact/misc/phenotype_annotation_hpoteam.tab`
†MESH Phenotype annotations	2019-08-15	.tab with tabular separator	`http://www.paccanarolab.org/static_content/disease_similarity/mim2mesh.tsv`
MESH Phenotype annotations (BioPortal)	2019-08-15	.txt file	`https://github.com/fair-workflows/openpredict/blob/master/data/external/meshAnnotationsFrom/BioPorttalUsingOMIMDesc.txt`

Table 1. All datasets used in OpenPREDICT version v0.1 and v0.2.

4.2 OpenPREDICT workflow representation

Figure 2 illustrates the main steps of the OpenPREDICT workflow, in which the Main Protocol is represented as a *dul:Workflow* and a *p-plan:Plan*, with version set through the *dc:hasVersion* property. The workflow consists of four steps: data preparation, feature generation, model training and evaluation, and presentation of results. Each one is defined by (*dul:isDescribedBy*) its own *p-plan:Plan*. In the first version of OpenPREDICT (0.1) all steps within the data preparation were manual (*bpmn:ManualTask*), as the FAIRification process and the preparation steps on data that were already provided as RDF. The second version of OpenPREDICT (0.2) automated most of these manual steps, requiring less human intervention. We will now go trough some of the most important aspects of this representation.

Prospective provenance. We decoupled the workflow steps from the instructions, linking a *p-plan:Step* to a *p-plan:Variable* through *p-plan:hasInputVar* and *p-plan:hasOutputVar*, while the *p-plan:Plan* links to the *prov:Usage* through the *prov:qualifiedUsage* property, describing how to bind the variable to other resources. This is an example:

```
opredict:Step_Download_Drugbank_dataset
  rdf:type bpmn:ManualTask ;
  rdf:type edam:operation_2409 ;
```

```
362    rdf:type p-plan:Step ;
363    p-plan:hasOutputVar opredict:Variable_Drugbank_dataset_online ;
364    p-plan:isStepOfPlan opredict:Plan_Main_Protocol_v01 ;
365    dul:isDescribedBy opredict:Plan_Download_Drugbank_dataset ;
366    dul:precedes opredict:Step_Save_Drugbank_dataset ;
367    rdfs:label "Download Drugbank dataset" ;
368  .
369
370  opredict:Plan_Download_Drugbank_dataset
371    rdf:type p-plan:Plan ;
372    dc:description "Download Drugbank dataset" ;
373    dc:language :LinguisticSystem_xsd_language_English ;
374    rdfs:label "Download Drugbank dataset" ;
375    prov:qualifiedUsage opredict:
376        Usage_Fetch_download_Drugbank_dataset_to_variable ;
377  .
378
379  opredict:Usage_Fetch_download_Drugbank_dataset_to_variable
380    rdf:type prov:Usage ;
381    rdfs:label "Link variable to download Drugbank dataset" ;
382    prov:entity opredict:Distribution_release-4-drugbank-drugbank.nq.gz;
383    prov:entity opredict:Variable_Drugbank_dataset_online ;
384  .
385
386  opredict:Distribution_release-4-drugbank-drugbank.nq.gz
387    rdf:type dcat:Distribution ;
388    rdfs:label "release/4/drugbank/drugbank.nq.gz" ;
389    dcat:downloadURL "http://download.bio2rdf.org/files/release/4/drugbank/
390        drugbank.nq.gz" ;
391    dcat:mediaType opredict:DataFormat_nq_compressed_gz ;
392  .
393
394  opredict:Variable_Drugbank_dataset_online
395    rdf:type p-plan:Variable ;
396    rdfs:label "Drugbank dataset online" ;
397  .
```

Retrospective provenance. We represent the concrete executions that happened and the concrete output that was generated with a *p-plan:Activity* that is linked to a *p-plan:Step* through the *p-plan:correspondsToStep* property and to the outputs (*opmw:WorkflowExecutionArtifact)* through *prov:generated*. Each output has a value (e.g., accuracy rate) and is linked to *prov:Generation* through the *prov:qualifiedGeneration* property, which specifies when the generation occurred with (*prov:atTime*). This is an example:

```
404  opredict:
405      Activity_Model_preparation_train_and_evaluation_Execution_1546302862
406    rdf:type p-plan:Activity ;
407    p-plan:correspondsToStep opredict:
408        Step_Model_preparation_train_and_evaluation ;
409    prov:generated opredict:ModelEvaluation_Accuracy_Execution_1546302862 ;
410    prov:generated opredict:
411        ModelEvaluation_AveragePrecision_Execution_1546302862 ;
412    prov:generated opredict:ModelEvaluation_F1_Execution_1546302862 ;
413    prov:generated opredict:ModelEvaluation_Precision_1546302862 ;
414    prov:generated opredict:ModelEvaluation_Recall_Execution_1546302862 ;
415    prov:generated opredict:ModelEvaluation_RocAuc_Execution_1546302862 ;
416  .
417
418  opredict:ModelEvaluation_Accuracy_Execution_1546302862
419    rdf:type mls:ModelEvaluation ;
```

```
420    dc:description "0.833336" ;
421    mls:specifiedBy opredict:EvaluationMeasure_PredictiveAccuracy ;
422    prov:qualifiedGeneration opredict:Generation_Execution_1546302862 ;
423    .
424
425  opredict:Generation_Execution_1546302862
426    rdf:type prov:Generation ;
427    prov:atTime "2019-01-01T00:02:31.011"^^xsd:dateTime ;
```

Versioning of workflows. We track the modification across the two versions with the *dc:hasVersion*
property on the level of a *dul:Workflow*, *p-plan:Plan*, *dc:LinguisticSystem*, and *prov:SoftwareAgent*.
Furthermore, we use the *prov:wasRevisionOf* property to link to the previous version. This is an example:

```
432  opredict:Plan_Main_Protocol_v02
433    rdf:type p-plan:Plan ;
434    rdf:type dul:Workflow ;
435    dc:created "2019-05-15" ;
436    dc:creator opredict:Agent_Remzi ;
437    dc:description "OpenPREDICT Main Protocol v.0.2" ;
438    dc:hasVersion "0.2" ;
439    dc:language :LinguisticSystem_xsd_language_English ;
440    dc:modified "2019-07-03" ;
441    pwo:hasFirstStep opredict:Step_Prepare_Input_Data_Files_v02 ;
442    rdfs:label "Main Protocol v.0.2" ;
443    prov:wasAttributedTo opredict:Agent_Remzi ;
444    prov:wasRevisionOf opredict:Plan_Main_Protocol_v01 ;
445    .
446
447  opredict:Plan_Main_Protocol_v01
448    rdf:type p-plan:Plan ;
449    rdf:type dul:Workflow ;
450    dc:created "2018-11-27" ;
451    dc:creator opredict:Agent_Remzi ;
452    dc:description "OpenPREDICT Main Protocol v.0.1" ;
453    dc:hasVersion "0.1" ;
454    dc:language :LinguisticSystem_xsd_language_English ;
455    dc:modified "2019-05-15" ;
456    pwo:hasFirstStep opredict:Step_Prepare_Input_Data_Files ;
457    rdfs:label "Main Protocol v.0.1" ;
458    prov:wasAttributedTo opredict:Agent_Remzi ;
459
460    .
```

5 EVALUATION

In this section we describe the evaluation of our approach, consisting of two parts: First, we revisit each
FAIR principle and explain how the principle is addressed. Second, we applied the traditional ontology
validation methodology by answering the competency questions through the execution of SPARQL queries
(the concrete queries are available in GitHub repo).

5.1 Addressing the FAIR principles

In order for our workflow to comply with FAIR principles, we checked each FAIR criterion defined in
(Wilkinson et al., 2016), as identified between parentheses below. First, global and persistent identifiers
were assigned to resources defined in the workflow and associated data. Rich metadata for workflow and
input and output data were created using HCLS [7] and FAIR data point specification (F2). In addition, the
metadata we generated contain an explicit global and persistent identifier of the data they describe (F3).

---

[7]https://www.w3.org/TR/hcls-dataset/

472 In order to enable the workflow and the data used to be searched, they were uploaded in a triple-store as
473 a FAIR Data Point [8]. Data can be queried through SPARQL over HTTP(S) protocol (A1.1). Since the
474 data is not private or protected, we don't require authentication and authorisation mechanism (A1.2). All
475 data and metadata are permanently available at Zenodo [9] to make the metadata accessible even the data is
476 no longer available (A2). We used RDF, OWL with commonly used controlled vocabularies, ontologies
477 such as Bio2RDF vocabulary, SIO and PROV to model input data and workflows (I1). HCLS dataset
478 specification and FAIR Data Point specification were used to define the metadata and provenance of data
479 (I2). Meaningful links between (meta)data such Bio2RDF links and data and workflow were created (I3).
480 We describe the semantic model of the workflow and the data with a commonly used vocabulary such as
481 ML-Schema and Reproduce-me (R1). We provide the license (R1.1) and provenance information in the
482 metadata using FAIR data point specification (R1.2), and HCLS specification (R1.3) and PROV.

5.2 Answering competency questions

484 Besides evaluating whether each FAIR principle was addressed, we also assessed the unified model
485 using the common semantic validation approach, which is based on SPARQL queries used to answer
486 the competency questions. All questions listed in Section 3 could be answered by running the SPARQL
487 queries over the OpenPREDICT use case. The complete queries and results obtained from OpenPREDICT
488 triplestore can be found in Github page. Therefore, the reproduction of this validation can be performed
489 by re-executing the queries on the RDF representation of the OpenPREDICT workflow. Below we explain
490 the result for each competency question.

5.2.1 CQ1 - Questions about manual steps.

492 **CQ1.1: Which steps are meant to be executed manually and which to be executed computation-**
493 **ally?** The SPARQL query we built to answer this question first filters all steps within the first version of
494 OpenPREDICT workflow (*opredict:Plan_Main_Protocol_v01*). The results show each step and its type -
495 manual (*bpmn:ManualTask*) or computational (*bpmn:ScriptTask*) - as well as the respective instructions
496 (*p-plan:Plan*) that describe the steps. In summary, OpenPREDICT v0.1 has 28 manual steps and 14
497 computational steps (42 in total), while v0.2 has 9 manual steps and 9 computational steps (18 in total).
498 This difference reflects the automatization of most of the manual steps within data preparation (evolving
499 from manual to computational) and the simplification of the computational steps described in fewer
500 Jupyter Notebook cells.

501 **CQ1.2:For the manual steps, who are the agents responsible to execute them?** To answer this
502 question we filtered the results for only manual steps through the statement:
503   *values ?stepType { bpmn:ManualTask }*
504   The result is a list of all steps and roles related to each one, such as executor, creator, developer, and
505 publisher. For example, Remzi is creator, developer and executor of all instructions, while Ahmed is
506 developer of some computational steps and Joao is the executor of the entire OpenPREDICT workflow.
507 This approach allows for the representation of multiple roles played by different agents within each step.
508   As in related approaches such as Workflow4ever and Reproduce-me, we use the PROV ontology to
509 address the different types of agents and roles through the *prov:wasAttributedTo* property, and apply the
510 *dc:creator* and *dc:publisher* properties for the direct relation from an instruction to an agent.

511 **CQ1.3: Which datasets were manually handled and what are their formats?** OpenPREDICT's
512 computational steps use datasets, as explained in Section 4.1, that required manual pre-processing. The
513 difference between v0.1 and v0.2 is that we automated the manual pre-processing of two datasets in
514 v0.2; MESH Phenotype annotations and protein-protein inter-actions. The main elements of the query
515 reflect the FAIR data point specification with DCAT elements (*dcat:Distribution, dcat:downloadURL*
516 and *dcat:mediaType*), PROV (*prov:Usage* and *prov:qualifiedUsage*) and EDAM classification for data
517 handling steps (*edam:operation_2409*) and data formats (media types).

518 **CQ1.4: What are the types of manual steps involved, and what are their inputs and outputs?**
519 Similar to the Reproduce-me approach, our ontology leverages on the P-PLAN ontology to address

---

[8]https://graphdb.dumontierlab.com/repositories/openpredict
[9]https://doi.org/10.5281/zenodo.3770918

the variables used as input and output of the manual steps, mostly during data preparation in Open-PREDICT v0.1, such as downloading and saving the datasets listed in the results of CQ1.3. For example, the input of *opredict:Step_Save_files_in_triplestore* are variables that indicate the local file of each dataset (serialized as RDF) and the output variable indicating the endpoint to upload all datasets (*opredict:Variable_Triplestore_endpoint_for_input_data*).

When changing the filter from manual steps to computational steps, the pattern followed was to classify the output variables of a step (a Jupyter Notebook cell) according to the data saved in files. For example, in feature generation, the *opredict:Step_Feature_generation_01_Pipeline_Source_Cell11* has an output variable for drug fingerprint similarity, indicating the generation of the file "drugs-fingerprint-sim.csv" .

5.2.2 CQ2 - Questions about instantiation of general workflows by more specific ones.

CQ2.1: What are the main steps of a general workflow? OpenPREDICT workflow follows the common machine learning pipeline process of: data preparation, feature generation, model training, model evaluation and presentation of results. The query returns these steps by looking for the first step of the workflow (through *pwo:hasFirstStep*) and following the preceding path in a recursive way, e.g.,

```
?step1 dul:precedes ?step2.
?step2 dul:precedes ?step3.
?step3 dul:precedes ?step4. (until there is no preceding steps)
```

The classification of the step is given by the EDAM specializations of the Operation concept *(operation_0004)*, such as *Data Handling* for data preparation (*edam:operation_2409*). For the sake of simplicity, model training and evaluation were performed within the same step. The main steps are listed below:

```
opredict:Step_Prepare_Input_Data_Files
opredict:Step_Feature_generation_Pipeline_OpenPREDICT_ipynb
opredict:Step_Model_preparation_train_and_evaluation_Workflow_OpenPREDCIT_-
    _ML_ipynb
opredict:Step_Format_results_for_presentation
```

CQ2.2: What are the steps of a specific workflow? Similar to the previous question, the SPARQL query uses the properties that allow for the ordering of steps' execution (*pwo:hasFirstStep* and *dul:precedes*). The pattern *p-plan:Step dul:isDescribedBy p-plan:Plan* allows us to answer this question, by representing how a step is described by an instruction. This pattern resembles the one used by Workflow4ever, which applies the *wfdesc:hasWorkflowDefinition* (*dul:isDescribed*) to link a *wfdesc:Workflow* (*p-plan:Step*) to a *wfdesc:WorkflowDefinition* (*p-plan:Plan*), aiming at representing the instructions (e.g., a Python script) that are natively understood by the *wfdesc:WorkflowEngine* (*prov:SoftwareAgent*). However, different from this approach, we classify the instruction language (*p-plan:Plan dc:language dc:LinguisticSystem*), allowing for the representation of instructions that follow computer language or natural language, which includes pseudo-code — commonly used to specify algorithms before implementing in a particular computer language.

The results show that OpenPREDICT has 78 steps in total, where 60 steps belong to v0.1 and 18 belong to v0.2, each step linked to an instruction. 9 instructions were reused from v0.1 to v0.2 regarding data preparation, thus, v0.2 presents 9 new instructions that are used to automate the data preparation phase. These instructions are written as either English (natural language) or Python 3.5 (computer language), where most of the Python ones refer to the Jupyter notebook cells for feature generation and model training and evaluation.

CQ2.3: What higher-level description does a certain workflow step instantiate? The SPARQL query to answer this question includes the pattern *p-plan:Plan dul:isDescribedBy p-plan:Plan*, which extends the capability described in the previous question, i.e. decoupling steps from instructions, enabling the representation of different abstraction levels of instructions and their relations. This pattern resembles the links between specification artefacts (e.g., conceptual model, activity diagrams and use cases) and implementation artefacts (e.g., software code, deployment procedures and automated tests) in software engineering. Usually, a specification artefact aims at describing the instructions necessary to enable a programmer to create the software code, sometimes automatically generated as in model-driven engineering. For example, a pseudo-code within an activity diagram (*p-plan:Plan*) may describe the behaviour

expected (*dul:isDescribed*) for the algorithm behind a service endpoint, which may be implemented as a Python script (*p-plan:Plan*).

OpenPREDICT did not formally followed the specification phase of software engineering since it is a research project, having the code developed from the data scientist interpretation perspective about publications related to PREDICT. In research-oriented data science this type of approach is common. However, we created some examples of the pattern that represent the specification of OpenPREDICT workflow. Therefore, the results of this query include 10 Jupyter Notebook cell instructions (*p-plan:Plan*), representing implementation artefacts, that were specified (p-plan:isDescribedBy) by 3 specification instructions (*p-plan:Plan*). The level of abstraction can be derived from the properties of the instruction. For example, the 10 Jupyter Notebook cell instructions were written (*dc:language*) in Python 3.5 (*schema:ComputerLanguage*), while the 3 specification instructions were written in English (*en* value of *xsd:language*). Furthermore, this approach enables links of *s* (specification artefacts) *x i* (implementation artefacts), where *i¿s*, i.e., a specification artefact usually describes several software code lines (instructions). In OpenPREDICT, the first specification instruction guides the load of input datasets, which is linked to cells 1-5 of the feature generation step, while the second guides the calculation of scores between pairs of drugs and compute similarity feature, which is linked to cells 6-9.

5.2.3 CQ3 - Questions about versioning of workflows and their executions

CQ3.1: What are the existing versions of a workflow and what are their provenance? The collective workflow (the whole) is represented as a *dul:Workflow* and a *p-plan:Plan*. Similar to other approaches (Workflow4ever, Reproduce-me, CWLProv, among others) the query to answer this question makes use of DC properties (e.g., *dc:creator, dc:created, dc:modified*) and PROV (e.g. *prov:wasAttributedTo*) for prospective provenance. It also covers workflow versioning through *dc:hasVersion* and *prov:wasRevisionOf*, where the former is responsible for version of *dul:Workflow* and the latter to link an instruction to another (*p-plan:Plan prov:wasRevisionOf p-plan:Plan* pattern). The retrospective (executions) provenance is supported by the link from an execution (a *p-plan:Activity*) to the correspondent step (*p-plan:correspondsToStep* property), which is a pattern that resembles most of the aforementioned semantic models. The main difference here is the assumption that any instruction (*p-plan:Plan*) should be versionable, thus, all executions link to a versioned instruction. Differently from Workflow4ever approach, here we do not introduce any elements regarding the specification of the changes (e.g., *roevo:ChangeSpecification*). The results for OpenPREDICT show 2 workflows (v0.1 and v0.2), both created by and attributed to Remzi, where v0.2 links to the prior version (v0.1).

CQ3.2: Which instructions were removed/changed/added from one version to another? Three SPARQL queries were written to answer whether the instructions of OpenPREDICT v0.1 were removed or changed or added in v0.2. Each SPARQL uses the identifier of the workflow versions (retrieved in CQ3.1) as an input parameter to perform the comparison from one version to another. For the query for removed instructions, it considers all instructions used in v0.1 that are not used in v0.2 and excludes the instructions that were changed. For the query for changed instructions, it considers the instructions with the *prov:wasRevisionOf* property. For the query for added instructions, the SPARQL query uses the reverse logic from the removed.

47 instructions were removed from v0.1 to v0.2 due to the refactoring of the code of feature generation, model training and model evaluation, and the elimination of several manual steps in data preparation. 3 instructions were changed, reflecting the porting of the FAIRification manual steps to computational steps in data preparation, i.e., download and save human interactome barabasi, and phenotype annotations. 7 instructions were added in v0.2, where 3 of them represent the new Python scripts for data preparation of the new data sources, other 3 represent the new scripts for feature generation and the remaining for model training.

CQ3.3: Which steps were automatized from one version to another? This query is quite similar to the one used for changed instructions (CQ3.2) but it makes explicit that the old version of the instruction used as manual step (*bpmn:ManualTask*) was modified to an instruction used as computational step (*bpmn:ScriptTask*) in the new version. The results confirm the findings from the previous query regarding the 3 instructions that were ported from manual steps to computational steps, namely the data preparation top-level instruction, the FAIRification instructions (download and save human interactome barabasi, and phenotype annotations). Although our approach covers change management, we face the same challenges regarding the dependency of the developer practices for code versioning. This means that, for example, a

626 developer is free to choose whether to remove files from an old version of the software implementation
627 and add files to the new version, even though these files refer to the same capability or routines. Most of
628 the version controls track the changes when the files (old and new) have the same name and path (i.e., the
629 step identifier), which is a similar approach used here.

630 **CQ3.4: Which datasets were removed, changed, oradded from one version to the next?** This
631 question can be answered by mixing the same query of CQ1.3 (datasets manually used) with the logic
632 used in query CQ3.2, i.e., one SPARQL query to the datasets removed, one for the changed and one for
633 the added. The query results over OpenPREDICT (v0.1 and v0.2) confirm the findings of CQ1.3, where
634 none datasets were removed from the old version to the new, none changed and 2 were added.

635 **CQ3.5: Which workflow version was used in each execution and what was generated?** This ques-
636 tion is answered by exploiting the pattern *p-plan:Activity p-plan:correspondsToStep p-plan:Step*, where
637 the step is part of the dul:Workflow that provides the workflow version. The OpenPREDICT workflow
638 had 14 executions represented with our unified model, exemplifying the execution of some computational
639 steps, i.e., each one a particular Jupyter Notebook cell. Therefore, this approach allows for the repre-
640 sentation of multiple executions of each step according to the version of the corresponding instruction.
641 Each execution inherits the properties of *p-plan:Activity*, e.g., the event start and end time points. Further-
642 more, each execution is associated to the correspondent generated artefacts through the *p-plan:Activity*
643 *prov:generated opmw:WorkflowExecutionArtifact* pattern, a similar approach of Workflow4ever, which
644 applied the inverse property *prov:wasGeneratedBy*. An artefact generated by an execution can be an eval-
645 uation measure of the model trained, such as the model accuracy and recall for that particular execution,
646 i.e., a *mls:ModelEvaluation*. Therefore, OpenPREDICT executions generated the values about the model
647 evaluation measures of accuracy, average precision, F1, precision, recall and ROC AUC. For example, the
648 results show that the model accuracy of v0.1 is 0.83, while v0.2 is 0.85.

649 This query can be further detailed by considering the particular version of each instruction that
650 the executed step implements. In addition, ideally, each output of a Jupyter Notebook cell should
651 be represented as a *opmw:WorkflowExecutionArtifact*, so all generated outputs are stored (similar to
652 ProvBook/Reproduce-me approach). This query can be easily changed to provide aggregations for related
653 analytical questions, such as how often each workflow version was executed.

6 DISCUSSION

6.1 Reproducibility Challenges

656 It was expected that our study would be unable to fully reproduce the accuracy of the method reported in
657 the PREDICT paper due to use of different input datasets. The performance results of this study are lower
658 than originally reported. The PREDICT paper reported an AUC of 0.90 in cross-validation, but using the
659 same gold standard, we could only achieved a lower AUC of 0.83.

660 We were also able to obtain the drug and disease similarity matrices used in PREDICT from the
661 authors via email request. Given 5 drug–drug similarity measures for 593 drugs and 2 disease–disease
662 similarity measures for 313 diseases, there are resulting 10 features of combined drug-disease similarities.
663 The logistic classifiers were trained with these pre-computed similarity scores and an average AUC of
664 0.85 was obtained from 10 repetitions in a 10-fold cross-validation scheme. This is still a significant
665 difference from the AUC of 0.90 what the authors reported in PREDICT study. This indicates that there
666 was more likely an error in the design or implementation of evaluation, not the aggregation data and
667 calculation of drug-drug and disease-disease similarity scores.

668 While attempting to reproduce the PREDICT study, we faced the following issues, which we then
669 turn into generic recommendations (highlighted in italics).

670 1. **Insufficient documentation** Essential details concerning the calculation of features were not
671 clearly defined, nor was the software code to perform the calculations provided. *Many details of*
672 *an experiment, including data sets, processing parameters and applied software and algorithms*
673 *need to be specified in order to facilitate the replication of the results. A methods section in a*
674 *scientific article may not be the best place to provide all this information as it is usually limited by*
675 *size constraints and different organization styles of journals and conference proceedings, leading*
676 *to a lack of required detail.*

2. **Inaccessible or missing data** Since no data except the gold standard data (drug–disease associations) were given, the features for the PREDICT were reconstructed using the publicly accessible databases DrugBank and KEGG and SIDER. However, we could not check if this resulted in exactly the same datasets. *The original data that were available to the authors could be absent or no longer accessible to others for many reasons. Sufficient data should be published to enable reproducing a study.*

3. **Versioning and change of data** In PREDICT, publicly accessible datasets have been used to construct models and validate hypotheses for prediction of drug indications and drugs were identified by their Drugbank IDs. However, Drugbank IDs are fully static and subject to change, and the original values may change throughout the time. For example, two drugs (DB00510, DB00313) in the original dataset were merged to the same drug within the current version of the Drugbank. *Results or hypotheses may change as a result of updating input data. In order to draw the same conclusion, it is important to record the version or the date of the data that were used in a study. This is especially important as publicly accessible datasets are increasingly used to construct models and validate hypotheses for prediction of drug indications.*

4. **Execution environment and third-party dependencies** In the PREDICT study, the versions of some software tools, such as the library for semantic similarity calculation, were not specified. *The versions of software libraries, packages and tools used in a workflow should be explicitly mentioned, and an effort must be made to maintain the access to those releases used in the original workflow.*

6.2 Issues of the semantic approach

While the execution of the FAIRification process in the OpenPREDICT was straightforward, the semantic modelling of the unified workflow model was challenging. The reuse of existing semantic vocabularies for the representation of our unified model showed to be an extensive task. There are several existing semantic approaches to represent workflows that present reproducibility issues and different conceptualizations, sometimes overlapping in their terminology. For example, the prospective part of Workflow4ever implementation (wfdesc[10]) has consistency issues such as missing disjointness and licensing elements, besides not conforming to the documentation (e.g., for all elements related to workflow templates). On the other hand, semantic models like DUL, PROV and P-PLAN presented higher quality and common foundations (in DOLCE), being easier to reuse and extend. Although CWLProv also provides an ontology based on W4ever semantic models, it is oriented only to retrospective provenance of computational steps, reusing most of the predicates that P-PLAN extends (from PROV). Furthermore, the URI (https://w3id.org/cwl/prov) does not provide a concrete description of the new predicates (e.g., cwlprov:image) and neither resolves to the RDF model (TBox).

A question that may arise is whether it would be better to create a new ontology from scratch rather than creating a unified model based on the existing ontologies. We believe that high quality semantic models should be reused, taking benefit from the lessons learned. Furthermore, we consider that reusing existing semantic workflow models really improve semantic interoperability, while creating a new ontology may impede interoperability if it is not accompanied with alignments to the existing semantic models. Therefore, our approach leads to an improved semantic interoperability. Because we reused several semantic models, the competency questions that they target are potentially addressed by our approach. For example, the gap in our approach regarding the representation of change management for versioning can be addressed by reusing some elements from the versioning approach of Workflow4ever, e.g., *roevo:Change*, *roevo:ChangeSpecification* and *roevo:VersionableResource*.

Deciding which type of approach should be used for role representation should be based on the needs for either a fine grained definition of the role/relator pattern (a reified relationship), such as the *prov:Association* approach, or a simple property, such as *dc:creator*. While the former (1) enriches the definition of the role (an improved representation capability), the later (2) is less verbose:

```
(1)
?Association prov:agent opredict:Remzi;
prov:hadRole opredict:Creator;
prov:hadPlan ?plan.
```

---

[10]https://raw.githubusercontent.com/wf4ever/ro/0.1/wfdesc.owl

```
729
730  (2)
731  ?plan dc:creator 'Remzi' .
```

One of the main challenges is to understand the different terminology used for similar conceptualizations. Although the definitions of terms like plan, process, protocol, procedure, workflow, plan specification and standard operating procedure seem to be the same (or quite overlapping), it is notorious that when they are used by different communities they carry specific semantics. How to grasp these differences is a crucial question that needs further exploration. For example, in the bioinformatics community, the term *Workflow* usually refer to an implemented (computational) piece of software, i.e., a set of programming language instructions, usually developed with a *Workflow Management System* as a *Workflow Application* (Da Cruz et al., 2012). On the other hand, in software engineering, the *Workflow* term is usually referred to a detailed business process within the Business Process Modelling (BPM) research. Usually, the BPM languages conform to graphical notations (e.g., BPMN, EPC, ARIS), targeted to human comprehension rather than computational ends (process design/modelling). On the other hand, some BPM languages focus on representing, in a lower-level of abstraction, the process execution details, e.g., BPEL (process implementation) (Rosemann and vom Brocke, 2015) . This is a topic extensively exploited by Service Oriented Architecture (SOA) initiatives. Several related work target the gap that exists between business process models and workflow specifications and implementations, such as service composition schemes (Stephan et al., 2012) and formal provenance of process executions and versioning (Krishna et al., 2019). Furthermore, some of these languages provide predicates for forks and conditionals, which were intentionally not included in the unified model since they have a high complexity - it is still a topic under discussion in the CWL community, for example.

In future work we will improve the modelling of manual steps by studying and, possibly, incorporating predicades from the SMART protocols ontology. We shall characterize the abstraction levels of workflows based on multi-level process modelling approaches, such as the widespread adopted APQC's Process Classification Framework (PCF). The PCF provides five abstraction levels for process specification, from a high abstraction level to detailed workflow specification: category (level 1), process group (level 2), process (level 3), activity (level 4) and task (level 5). Although this framework aims at providing a methodological approach for business process specification, we should investigate whether the minimal information elements of each level require proper representation in the ontology. We should also consider that challenges of process refinement ("process description in a more fine-grained representation") (Ren et al., 2013) . A process refinement mechanism maps and/or derives models from a higher level specification to a detailed level, equivalent to vertical and exogenous model transformations in model-driven engineering. Typical refinement categories will be investigated, such as activity decomposition principles about event delivering and execution condition transferring (Jiang et al., 2016; Muehlen and Rosemann, 2004). The representation of intentionality of the activities within business processes will also be addressed in future work through goal-oriented semantic process modeling (Horkoff et al., 2019), linking goals to activities and roles.

Industry-oriented approaches are also being investigated, such as Extract, Transform and Loading (ETL/ELT) for data warehousing, such as SQL Server Integration Services, which considers a workflow as a control flow, while source to destination flows of data are performed by data flows. Furthermore, Product Line Management (PLM) tools should be investigated, especially the ones that cover Laboratory Information Management System (LIMS), which provides important concepts such as Bill-of-Materials (BoM), specifications and their certifications. For example, in PLM a *specification* is a description of raw materials and packaging materials, and semi-finished and finished products. This description may contain product characteristics (e.g., chemical compounds), recipes (e.g., BoM), production methods, quality norms and methods, artwork, documents, among others.

Ultimately, initiatives like CWL, the Center for Expanded Data Annotation and Retrieval (CEDAR) (for metadata management) (Gonçalves et al., 2017) and FAIRsharing.org (for indexing FAIR standards) may be used as building blocks for the envisioned FAIR workbench tool, which can be a reference implementation over a workflow system such as Jupyter Notebook (e.g., a plug-in). Finally, the validation of the reproducibility level of a workflow should consider specific FAIR metrics that takes in consideration specific recommendations (e.g., from CWLProv approach) and the practices for higher reproducibility of Jupyter notebooks (Pimentel et al., 2019).

7 CONCLUSIONS

In this work, we examined how the FAIR principles can be applied to scientific workflows. We adopted the FAIR principles to make the PREDICT workflow, a drug repurposing workflow based on machine learning, open, reproducible, and interoperable. Therefore, the main contribution of this paper is the OpenPREDICT case study, which demonstrates how to make a machine learning workflow FAIR and open. For this, we have created an unified model that reuses several semantic models to show how a workflow can be semantically modeled. We published the workflow representation, data and meta-data in a triple store which was used as FAIR data point. In addition, new competency questions have been defined for FAIR workflows and how these questions can be answered through SPARQL queries. Among the main lessons learned, we highlight how the main existing workflow modelling approaches can be reused and enhanced by our unified model. However, reusing these semantic models showed to be a challenging task, once they present reproducibility issues and different conceptualizations, sometimes overlapping in their terminology.

For the future, we envision that the intensive human effort that we had to perform in order to make a workflow FAIR will be taken care of by smart and intuitive workflow tools. As a prototype of such a tool, we are currently developing the FAIR workbench as a general tool that allows users to deal with workflows and protocols in a semantic and FAIR form.

8 ACKNOWLEDGMENTS

This research was funded by the NWO (grant number 628.011.011) and the Netherlands eScience Center (grant number P 17.0201).

REFERENCES

Ashburn, T. T. and Thor, K. B. (2004). Drug repositioning: identifying and developing new uses for existing drugs. *Nature Reviews Drug Discovery*, 3(8):673–683.

Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454.

Barrell, D., Dimmer, E., Huntley, R. P., Binns, D., O'Donovan, C., and Apweiler, R. (2009). The goa database in 2009—an integrated gene ontology annotation resource. *Nucleic acids research*, 37(suppl_1):D396–D403.

Barrett, T. and Edgar, R. (2006). *[19] Gene Expression Omnibus: Microarray Data Storage, Submission, Retrieval, and Analysis*, volume 411, pages 352–369. Academic Press.

Begley, C. G. and Ellis, L. M. (2012). Drug development: Raise standards for preclinical cancer research. *Nature*, 483(7391):531–533.

Belhajjame, K., Zhao, J., Garijo, D., Gamble, M., Hettne, K., Palma, R., Mina, E., Corcho, O., Gómez-Pérez, J. M., Bechhofer, S., Klyne, G., and Goble, C. (2015). Using a suite of ontologies for preserving workflow-centric research objects. *Journal of Web Semantics*, 32:16–42.

Bisgin, H., Liu, Z., Fang, H., Kelly, R., Xu, X., and Tong, W. (2014). A phenome-guided drug repositioning through a latent variable model. *BMC bioinformatics*, 15(1):267–267.

Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22.

Borgo, S. and Masolo, C. (2010). *Ontological Foundations of dolce*, pages 279–295. Springer Netherlands, Dordrecht.

Callahan, A., Cruz-Toledo, J., and Dumontier, M. (2013). Ontology-based querying with bio2rdf's linked open data. In *Journal of biomedical semantics*, volume 4, page S1. BioMed Central.

Caniza, H., Romero, A. E., and Paccanaro, A. (2015). A network medicine approach to quantify distance between hereditary disease modules on the interactome. *Scientific reports*, 5:17658.

Cheng, F., Liu, C., Jiang, J., Lu, W., Li, W., Liu, G., Zhou, W., Huang, J., and Tang, Y. (2012). Prediction of drug-target interactions and drug repositioning via network-based inference. *PLOS Computational Biology*, 8(5):e1002503.

Cohen-Boulakia, S., Belhajjame, K., Collin, O., Chopard, J., Froidevaux, C., Gaignard, A., Hinsen, K., Larmande, P., Bras, Y. L., Lemoine, F., Mareuil, F., Ménager, H., Pradal, C., and Blanchet, C. (2017). Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. *Future Generation Computer Systems*, 75:284–298.

Collins, S., Genova, F., Harrower, N., Hodson, S., Jones, S., Laaksonen, L., Mietchen, D., Petrauskaitė, R., and Wittenburg, P. (2018). Turning fair into reality: Final report and action plan from the european commission expert group on fair data.

Correa Publio, G., Esteves, D., Lawrynowicz, A., Panov, P., Soldatova, L., Soru, T., Vanschoren, J., and Zafar, H. (2018). Ml-schema: Exposing the semantics of machine learning with schemas and ontologies. In *Reproducibility in Machine Learning Workshop, ICML*.

CrowdFlower (2016). Data science report. Report. accessed on 09 October 2019.

Da Cruz, S. M. S., Campos, M. L. M., and Mattoso, M. (2012). A foundational ontology to support scientific experiments. *CEUR Workshop Proceedings*, 938(2001):144–155.

Garijo, D. and Gil, Y. (2012). Augmenting prov with plans in p-plan: Scientific processes as linked data. In *LISC@ISWC*.

Giraldo, Olga, G. A. L. F. C. O. (2017). Using semantics for representing experimental protocols. *Journal of Biomedical Semantics*.

Gonçalves, R. S., O'Connor, M. J., Martínez-Romero, M., Egyedi, A. L., Willrett, D., Graybeal, J., and Musen, M. A. (2017). The cedar workbench: An ontology-assisted environment for authoring metadata that describe scientific experiments. *The Semantic Web – ISWC 2017*, page 103–110.

Gottlieb, A., Stein, G. Y., Ruppin, E., and Sharan, R. (2011). Predict: a method for inferring novel drug indications with application to personalized medicine. *Molecular Systems Biology*, 7(1):496.

Gray, K. A., Yates, B., Seal, R. L., Wright, M. W., and Bruford, E. A. (2015). Genenames. org: the hgnc resources in 2015. *Nucleic acids research*, 43(D1):D1079–D1085.

Guizzardi, G., Wagner, G., Almeida, J. P. A., and Guizzardi, R. S. S. (2015). Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. *Applied Ontology*, 10(3-4):259–271.

Hartanto, H. A., Sarno, R., and Ariyani, N. F. (2017). Warning criterion ontology for measuring of compliance in standard operating procedure implementation.

Hettne, K., Wolstencroft, K., Belhajjame, K., Goble, C., Mina, E., Dharuri, H., Verdes-Montenegro, L., Garrido, J., De Roure, D., and Roos, M. (2012). Best practices for workflow design: How to prevent workflow decay. *CEUR Workshop Proceedings*, 952.

Hoehndorf, R., Hiebert, T., Hardy, N. W., Schofield, P. N., Gkoutos, G. V., and Dumontier, M. (2013). Mouse model phenotypes provide information about human drug targets. *Bioinformatics*, 30(5):719–725.

Horkoff, J., Aydemir, F. B., Cardoso, E., Li, T., Maté, A., Paja, E., Salnitri, M., Piras, L., Mylopoulos, J., and Giorgini, P. (2019). Goal-oriented requirements engineering: an extended systematic mapping study. *Requirements Engineering*, 24(2):133–160.

Imming, M., Böhmer, J., Companjen, B., Emery, T., Groep, D., Murchison, K., Schoonhoven, R., Sesink, L., Som de Cerff, W., Sterl, A., and Franke, W. (2018). Fair data advanced use cases: from principles to practice in the netherlands.

Ioannidis, J. P. A. (2005a). Contradicted and initially stronger effects in highly cited clinical research. *JAMA*, 294(2):218–228.

Ioannidis, J. P. A. (2005b). Why most published research findings are false. *PLOS Medicine*, 2(8):e124.

Jacobsen, A., Kaliyaperumal, R., da Silva Santos, L. O. B., Mons, B., Schultes, E., Roos, M., and Thompson, M. (2019). A generic workflow for the data fairification process. *Data Intelligence*, pages 56–65.

Jiang, Y., Xiao, N., Zhang, Y., and Zhang, L. (2016). A novel flexible activity refinement approach for improving workflow process flexibility. *Computers in Industry*, 80:1–15.

Kanehisa, M., Araki, M., Goto, S., Hattori, M., Hirakawa, M., Itoh, M., Katayama, T., Kawashima, S., Okuda, S., Tokimatsu, T., and Yamanishi, Y. (2007). KEGG for linking genomes to life and the environment. *Nucleic Acids Research*, 36(Database):D480–D484.

Khan, F. Z., Soiland-reyes, S., Sinnott, R. O., Lonie, A., Goble, C., and Crusoe, M. R. (2019). Sharing interoperable work ow provenance : A review of best practices and their practical application in cwlprov. *GigaScience*, pages 1–26.

Klein, R. A., Ratliff, K. A., Vianello, M., Adams Jr, R. B., Bahnǐk, v., Bernstein, M. J., Bocian, K., Brandt, M. J., Brooks, B., Brumbaugh, C. C., Cemalcilar, Z., Chandler, J., Cheong, W., Davis, W. E., Devos, T., Eisner, M., Frankowska, N., Furrow, D., Galliani, E. M., Hasselman, F., Hicks, J. A., Hovermale, J. F., Hunt, S. J., Huntsinger, J. R., Ijzerman, H., John, M.-S., Joy-Gaba, J. A., Barry Kappes, H., Krueger,

L. E., Kurtz, J., Levitan, C. A., Mallett, R. K., Morris, W. L., Nelson, A. J., Nier, J. A., Packard, G., Pilati, R., Rutchick, A. M., Schmidt, K., Skorinko, J. L., Smith, R., Steiner, T. G., Storbeck, J., Van Swol, L. M., Thompson, D., van 't Veer, A. E., Vaughn, L. A., Vranka, M., Wichman, A. L., Woodzicka, J. A., and Nosek, B. A. (2014). Investigating variation in replicability: A "many labs" replication project. *Social Psychology*, 45(3):142–152.

Krishna, A., Poizat, P., and Salaün, G. (2019). Checking business process evolution. *Science of Computer Programming*, 170:1–26.

Kuhn, M., Campillos, M., Letunic, I., Jensen, L. J., and Bork, P. (2010). A side effect resource to capture phenotypic effects of drugs. *Molecular systems biology*, 6(1).

Lamb, J., Crawford, E. D., Peck, D., Modell, J. W., Blat, I. C., Wrobel, M. J., Lerner, J., Brunet, J.-P., Subramanian, A., Ross, K. N., Reich, M., Hieronymus, H., Wei, G., Armstrong, S. A., Haggarty, S. J., Clemons, P. A., Wei, R., Carr, S. A., Lander, E. S., and Golub, T. R. (2006). The connectivity map: Using gene-expression signatures to connect small molecules, genes, and disease. *Science*, 313(5795):1929–1935.

Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E., Dominguez Del Angel, V., van de Sandt, S., Ison, J., Martinez, P. A., McQuilton, P., Valencia, A., Harrow, J., Psomopoulos, F., Gelpi, J. L., Chue Hong, N., Goble, C., and Capella-Gutierrez, S. (2019). Towards FAIR principles for research software. *Data Science*, pages 1–23.

Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., and Zhao, J. (2013). Prov-o: The prov ontology. *W3C recommendation*, 30.

Menche, J., Sharma, A., Kitsak, M., Ghiassian, S. D., Vidal, M., Loscalzo, J., and Barabási, A.-L. (2015). Uncovering disease-disease relationships through the incomplete interactome. *Science*, 347(6224):1257601.

Moher, D., Liberati, A., Tetzlaff, J., Altman, D. G., and The, P. G. (2009). Preferred reporting items for systematic reviews and meta-analyses: The prisma statement. *PLOS Medicine*, 6(7):e1000097.

Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., and Paulson, P. (2008). The open provenance model: An overview. In *International Provenance and Annotation Workshop*, pages 323–326. Springer.

Moreira, J. L. R., Sales, T. P., Guerson, J., Braga, B. F. B., Brasileiro, F., and Sobral, V. (2016). Menthor editor: An ontology-driven conceptual modeling platform. In *JOWO@FOIS*.

Muehlen, M. z. and Rosemann, M. (2004). Multi-paradigm process management. In *CAiSE Workshops*, pages 169–175.

Neil, C. H. and Daniel S., K. (2018). *FAIR enough? Can we (already) benefit from applying the FAIR data principles to software?*

Noy, N. F., Shah, N. H., Whetzel, P. L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D. L., Storey, M.-A., Chute, C. G., and Musen, M. A. (2009). BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 37(Web Server):W170–W173.

Pimentel, J. F., Murta, L., Braganholo, V., and Freire, J. (2019). A large-scale study about quality and reproducibility of jupyter notebooks. In *Proceedings of the 16th International Conference on Mining Software Repositories*, pages 507–517. IEEE Press.

Poldrack, R. A. and Poline, J.-B. (2015). The publication and reproducibility challenges of shared data. *Trends in Cognitive Sciences*, 19(2):59–61.

Prinz, F., Schlange, T., and Asadullah, K. (2011). Believe it or not: how much can we rely on published data on potential drug targets? *Nature Reviews Drug Discovery*, 10(9):712–712.

Ren, Y., Gröner, G., Lemcke, J., Rahmani, T., Friesen, A., Zhao, Y., Pan, J. Z., and Staab, S. (2013). Process refinement validation and explanation with ontology reasoning. Service-Oriented Computing, pages 515–523. Springer Berlin Heidelberg.

Rosemann, M. and vom Brocke, J. (2015). *The Six Core Elements of Business Process Management*, pages 105–122. Springer Berlin Heidelberg, Berlin, Heidelberg.

Rospocher, M., Ghidini, C., and Serafini, L. (2014). An ontology for the business process modelling notation. In *FOIS*.

Samuel, S. and König-Ries, B. (2018a). Combining p-plan and the reproduce-me ontology to achieve semantic enrichment of scientific experiments using interactive notebooks. In *European Semantic Web Conference*, pages 126–130. Springer.

Samuel, S. and König-Ries, B. (2018b). Provbook: Provenance-based semantic enrichment of interactive notebooks for reproducibility. In *In Proceedings of the ISWC 2018 Posters Demonstrations, Industry*

944   *and Blue Sky Ideas Tracks co-located with ISWC.*

945   Scannell, J. W., Blanckley, A., Boldon, H., and Warrington, B. (2012). Diagnosing the decline in

946   pharmaceutical r&d efficiency. *Nature Reviews Drug Discovery*, 11(3):191–200.

947   Sirota, M., Dudley, J. T., Kim, J., Chiang, A. P., Morgan, A. A., Sweet-Cordero, A., Sage, J., and Butte,

948   A. J. (2011). Discovery and preclinical validation of drug indications using compendia of public gene

949   expression data. *Science Translational Medicine*, 3(96):96ra77–96ra77.

950   Sleigh, S. H. and Barton, C. L. (2010). Repurposing strategies for therapeutics. *Pharmaceutical Medicine*,

951   24(3):151–159.

952   Soiland-Reyes, S., Khan, F. Z., Sinnott, R., Lonie, A., Crusoe, M. R., and Goble, C. (2018). Capturing

953   interoperable reproducible workflows. In *Workshop on Research Objects: Workshop at IEEE eScience*

954   *2018*.

955   Stephan, B., Thomas, B., and Manfred, R. (2012). *Bridging the Gap between Business Process Models*

956   *and Service Composition Specifications*, pages 124–153. IGI Global, Hershey, PA, USA.

957   Vasilevsky, N. A., Brush, M. H., Paddock, H., Ponting, L., Tripathy, S. J., Larocca, G. M., and Haendel,

958   M. A. (2013). On the reproducibility of science: unique identification of research resources in the

959   biomedical literature. *PeerJ*, 1:e148–e148.

960   Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N.,

961   Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas,

962   M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G.,

963   Groth, P., Goble, C., Grethe, J. S., Heringa, J., 't Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J.,

964   Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van

965   Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson,

966   M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft,

967   K., Zhao, J., and Mons, B. (2016). The fair guiding principles for scientific data management and

968   stewardship. *Nature*, 3:160018.

969   Wishart, D. S., Knox, C., Guo, A. C., Cheng, D., Shrivastava, S., Tzur, D., Gautam, B., and Hassanali, M.

970   (2008). Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic acids research*,

971   36(suppl_1):D901–D906.

972   Wu, C., Gudivada, R. C., Aronow, B. J., and Jegga, A. G. (2013). Computational drug repositioning

973   through heterogeneous network clustering. *BMC systems biology*, 7 Suppl 5(Suppl 5):S6–S6.