Repeatability and Re-usability in Scientific Processes: Process Context, Data Identification and Verification

©Andreas Rauber, ©Tomasz Miksa, ©Rudolf Mayer, © Stefan Proell

SBA Research, and
Vienna University of Technology
Vienna, Austria
{arauber, tmiksa, rmayer, sproell} @sba-research.org

Abstract

eScience offers huge potential of speeding up scientific discovery, being able to flexibly re-use, combine and build on top of existing tools and results. Yet, to reap the benefits we must be able to actually perform these activities, i.e. having the data, processing components etc. available for redeployment and being able to trust them. Thus, repeatability of e-Science experiments is a requirement of validating work to establish trust in results. This proves challenging as procedures currently in place are not set up to meet these goals.

Several approaches have tackled this issue from various angles. This paper reviews these building blocks and ties them together. It starts from the capture and description of entire research processes and ways to document them. Regarding data, we review the recommendations of the Research Data Alliance on how to precisely identify arbitrary subsets of potentially high-volume and highly dynamic data used in a process. Last, we present mechanisms for verifying the correctness of process reexecutions.

1 Introduction

New means of performing research and sharing results offers huge potential for speeding up scientific discovery, enabling scientists to flexibly re-use, combine and build on top of results without geographical or time limitations and across discipline boundaries. Yet, to reap the benefits promised by eScience [13], we must be able to actually perform these activities, i.e. having the data, processing components available for re-deployment. Funding agencies such as the EC [1] are committed to data re-use and open data initiatives. As a result, all research data from publicly funded projects needs to be made available for the public. Not only does this entail that the data must be equipped with useful and stable metadata, comprehensive descriptions and documentation, but also that the data must be preserved for the long term. Yet, from an eScience perspective, mere availability of data is not sufficient, as data as such is barely useful. First of all, eScience benefits not only from the availability of data, but also from the re-use and re-purposing of tools and entire experimental workflows. Secondly, and more importantly, data never exists solely on its own, but is usually the result of more or less complex (pre-)processing chains. This commences with the processing happening at a sensor level or other processing happening during data capture, via analysis processes resulting in processed data, leading up to experimental results serving as input for further meta-studies. Thus, in both cases, we need to ensure that we have the underlying tools and processes available. This is necessary to understand their impact on the result, potential bias introduced by them, and to apply identical processing to new data to ensure comparability of results. To re-use such processing tools we need to trust them and any underlying components to produce identical (comparable) results under identical (similar) conditions.

From a scientific point of view, the validation of such research results (or, in fact, the result of every individual processing step) is a core requirement needed for establishing such trust in the scientific community, its tools and data, specifically in dataintensive domains. This proves challenging as procedures currently in place are not set up to meet these goals. Experiments are often complex chains of processing, involving a number of data sources, computing infrastructure, software tools, or external and third-party services, all of which are subject to change dynamically. In scientific research external

[1] ec.europa.eu/digital-agenda/en/open-data-0

Proceedings of the XVII International Conference «Data Analytics and Management in Data Intensive Domains» (DAMDID/RCDL'2015), Obninsk, Russia, October 13 - 16, 2015

influences can have a large impact on the outcome of an experiment. Human factors, the used tools and equipment, the configuration of soft- and hardware, the execution environment and its properties are important factors which need to be considered. The impact of such dependencies has proven to be graver than expected. While many approaches rely on documenting the individual processing steps performed during an experiment, on storing the data as well as the code used to perform an analysis, the impact of the underlying software and hardware stack are often ignored. Yet, beyond the challenges posed by the actual experiment/analysis, it is the complexity of the computing infrastructure (both the processing workflows and their dependencies on HW and SW environments, as well as the enormous amounts of data being processed) that renders research results in many domains hard to verify. As a recent study in the medical domain has prominently shown [11], even assumed minute differences such as the specific version of the operating system used can have a massive impact: different results were obtained in cortical thickness and volume measurements of neuroanatomical structures if the software setup of FreeSurfer, a popular software package processing MRI scans, is varied. More dramatically, though, there was also a difference in the result if not the primary software, but only the operating system versions (in this case the Mac-OSX 10.5 and 10.6) differ. This indicates the presence of dependencies from FreeSurfer to functions provided by the operating system, causing instabilities and misleading results. As these dependencies are hidden from the physician, such side-effects of the ICT infrastructure need to be detected and resolved transparently if we want to be able to trust results based on computational analyses.

A number of approaches have tackled this issue from various angles, including initiatives for data sharing, code versioning and publishing as open source, the use of workflow engines to formalize the steps taken in an experiment, to ways to describe the complex environment an experiment is executed in. In addition the data that is created but also the processing algorithms, scripts, and other software tools used in the experiment need to be accessible for longer time periods, for facilitating data reuse and allowing peers to retrieve and verify experiments. Keeping these assets accessible is not only a technical challenge, but requires institutional commitment and defined procedures.

Repeatability and reproducibility are two fundamental concepts in science. An experiment is repeatable, if it produces the exact same results under the very same preconditions. An experiment is reproducible, if the same results can be obtained even under somewhat different conditions, e.g. performed by a different team in a different location. There are several factors which have an influence on the variance of experiments. The ISO standard 57251:1994 [14] lists the following factors: (1) operator, (2) equipment, (3) calibration of the equipment, (4) environment and (5) time elapsed between measurements. The standard defines an experiment as repeatable, if the mentioned influences (1) - (4) are constant and (5) is a reasonable time span between two executions of the experiment and its verification. Reproducibility allows variance in these factors, as they cannot be avoided if different research teams want to compare results.

To tackle these issues we proposed to introduce Process Management Plans (PMPs) [23]. They extend Data Management Plans by taking a process centric view, viewing data simply as the result of underlying processes such as capture, (pre-) processing, transformation, integration and analyses. The general objective of PMPs is to foster identification, description, sharing and preservation of scientific processes. To embody the concept of PMPs we need to solve the challenges related to the description of computational processes, verification and validation, monitoring external dependencies, as well as data citation. This paper reviews these building blocks and ties them together to demonstrate the feasibility of sharing and preservation of not only datasets, but also scientific processes.

Section 2 summarizes related work from the areas of Data Management Plans (describing the result of data capturing/production processes), digital preservation of processes, and several eScience research infrastructures. Section 3 presents the Context Model that is automatically captured, describing the process implementation including all software and hardware dependencies. Ways to precisely identify and cite arbitrary subsets of dynamic data are described in Section 4, presenting the recommendations of the RDA WG on Data Citation. Section 5 discusses the verification and validation of the reexecution of computational processes. These concepts are illustrated via a use case from the machine learning domain in Section 6, followed by conclusions in Section 7.

2 Related Work

2.1 Data Management Plans

A prominent reason for the non-reproducibility of scientific experiments is poor data management, as criticized in several disciplines. Different data sets scattered around different machines with no track of dependency between them are a common landscape for particle physicists who move quickly from one research activity to another [5]. Several institutions reacted, publishing templates and recommendations for DMPs, such as the Digital Curation Centre (DCC) [9], Australian National Data Services (ANDS) [3] and National Science Foundation (NSF) [24], amongst many others. These are very similar, containing a set of advises, mainly lists of questions which researches should consider when developing a DMP. The attention is attracted to what

happens with data after it has been created, rather than in what way it was obtained. All the description is provided in a text form, and in case of NSF there is a limit of 2 pages. Thus, it is unlikely anybody will be able to reuse or at least reproduce the process which created the data. Furthermore, the correctness of data is taken for granted and thus DMPs do not provide sufficient information that would allow validating the data. Finally, the quality and detail of information strongly depends on the good will of researchers. There is no formal template for specification of DMPs which would ensure that all important information is covered comprehensively. Several tools are available, like *DMPonline*[2] for DCC or *DMPtool*[3] for NSF, which aid the researcher in the process of DMP creation, but they are rather simple interactive questionnaires which generate a textual document at the end, rather than the complex tools required to validate at least the appropriateness of the provided information. The main conclusion from the analysis is that DMPs focus on describing results of experiments. This is a consequence of their data centric view, which enforces focus on access and correct interpretation (metadata) of data and does not pay much attention to processing of data. While these constitute a valuable step in the right direction, we need to move beyond this, taking a process centric view.

2.2 Digital Preservation

The area of digital preservation is shifting focus from collections of simple objects to the long term preservation of entire processes and workflows.

WF4Ever[4] addressed the challenges of preserving scientific experiments by using abstract workflows that are reusable in different execution environments [26]. The abstract workflow specifies conceptual and technology-independent representations of the scientific process. They further developed new approaches to share workflows by using an RDF repository and make the workflows and data sets accessible from a SPARQL Endpoint[10]. The TIMBUS [4] project addressed the preservation of business processes by ensuring continued access to services and software necessary to properly render, validate and transform information. The approach centers on a context model [20] of the process, which is an ontology for describing the process components and their dependencies. It allows to store rich information, ranging from software and hardware to organizational and legal aspects. The model can be used to develop preservation strategies and redeploy the process in a new environment in the future. The project developed a verification and validation method for redeployed processes [12] that evaluates the conformance and performance quality processes redeployed in new environments. This is important when we want to reuse it to build other processes.

2.3 eScience and Research Infrastructures

Several projects benefit nowadays from sharing and reusing data [6]. In [7] the evolution of research practices by sharing of tools, techniques and resources is discussed. *myExperiment* [31] is a platform for sharing scientific workflows. This is already one step beyond just sharing the data. Workflows created and run within the *Taverna* workflow engine can be published and reused by other researchers. However, the workflows do not always specify all required information (e.g. tools to run the steps, description of parameters) to re-run the workflow [19].

An environment which enables scientists to collaboratively conduct their research and publish it in form of executable paper was presented in [25]. The solution requires working in a specific environment, limiting its applicability to the tools and software supported by the environment. PMPs does not have such a requirement and can be used in every case. There is a strong move towards ”providing a consistent platform, software and infrastructure, for all users in the European Research Area to gain access to suitable and integrated computing resources” [2].

3 Documenting eScience Processes

To enable analysis, repeatability and reuse of processes, they must be well described and documented. As most processes are rather complex in their nature, a precise description is needed to re-enact the execution of the process. Thus, formalized models are useful for a detailed representation of critical aspects such as the hardware, software, data and execution steps supporting the process, as well as their relationships and dependencies. Several models can be considered for this type of documentation.

Workflow-Centric Research Objects [15] (ROs) are a means to aggregate or bundle resources used in a scientific investigation, such as a workflow, provenance from results of its execution, and other digital resources such as publications, data-sets. In addition, annotations are used to further describe these digital objects. The model of Research Objects is in the form of an OWL ontology, and incorporates several existing ontologies. At its core, the Research Object model extends the *Object Exchange and Reuse* model (ORE) [33][5] to formalize the aggregation of digital resources. Annotations are realized by using the Annotation Ontology (AO) [4], which allows e.g. for comment and tag-style textual annotations. Specifying the structure of an abstract workflow is enabled by the *wfdesc* ontology. Finally, the provenance of a specific execution of a workflow is described using the *wfprov* ontology. Research objects have also been

[2] dmponline.dcc.ac.uk/
[3] dmp.cdlib.org/ 4 wf4ever-project.org/
[4] http://timbusproject.net/
[5] openarchives.org/ore/1.0

presented as a means to preserve scientific processes [8], proposing archiving and autonomous curation solutions that would monitor the decay of workflows.

Enterprise architecture (EA) modelling languages provide a holistic framework to describe several aspects of a process. For example, the Archimate [30] language supports description, analysis and visualization of the process architecture, on three distinct but interrelated layers: business, application and technology layer. On each of these layers, active structures, behavior and passive structures can be
modelled. Thus the process can be specified not only as a high level sequence of steps, but also as a low

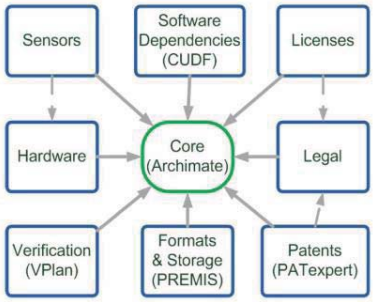


Fig. 1 Overview on the Context Model architecture: cor and extensions

level sequence of inputs and outputs from software and hardware components needed to run the process, e.g. database software, libraries, software device drivers, fonts, codecs, or dedicated hardware created for the purpose of the experiment. Enterprise architectures do not address any specific domaindependent concerns. They rather cut across the whole organization running the process [16].

While models such as Archimate or Research Objects are extensive, they often do not provide enough detail on technology aspects of the process, and thus in these aspects provide only little guidance to researchers aiming to produce a solid description of their technical infrastructure. One approach to alleviate this issue is realized in the Process Context Model [17], which builds on top of Archimate and extends it with domain specific languages to address specific requirements of a given domain. Wherever possible, the extension ontologies are based on already existing languages. The development of the model was driven by requirements to preserve and re-execute complete processes. The context a process is embedded in covers immediate and local aspects such as the software and hardware supporting the process, to aspects such as the organization the process is executed in, the people involved, service providers, to even laws and regulations. The exact context can differ significantly depending on the domain the process stems from.

The model is using the domain-independent Archimate language as a core model to integrate the domain specific extension languages. It is implemented in the Web Ontology Language (OWL) [34], and the integration is performed via ontology mapping from the extensions to the core model. An overview of this architecture and the provided domain-specific extensions is given in Figure 1, consisting of:

Software Dependencies cover dependencies between different types of software, including information on which versions are compatible or conflicting with each other. It is, for example, important to know that a specific version of a Java Virtual Machine is required to run a certain piece of software, or that a particular application is required to view a digital object. This is important when considering preservation of specific parts of the software stack utilized in the process. Beyond repeatability, this information may be used during preservation planning to identify alternative software applications that can be utilized. Technical dependencies on software and operating systems in the Context Model can be captured and described via the Common Upgradeability Description Format (CUDF) [32].

Data Formats In a process execution, a number of digital objects are created, modified or read. This section includes information on which data/file formats these are stored in. It is used for preservation actions and for selecting appropriate comparator modules during the validation process described in Sec. 5 Our implementation of the Context Model uses the PREMIS Data Dictionary [27] to represent this information.

Hardware contains a comprehensive description of the computational hardware, from desktop systems, server infrastructure components, to specialized hardware used for certain tasks. Even though in many processes the hardware employed to host the software applications might be standard commodity hardware, its exact specifications can still influence the run-time behavior of a process. This might be critical in certain circumstances, such as execution speed, or when specific functionalities and characteristics of the hardware such as precision limits, analog/digital conversion thresholds etc. are part of the computation. Further, certain processes might use certain hardware capabilities for computation, such as using graphical processing units (GPUs) for large-scale experiments in scientific processes. These types of hardware, and the software that can utilize them, are not yet as standardized and abstracted, thus an exact description is needed in many cases.

Legal aspects cover legal restrictions imposed on the processes. *License* information focusies specifically on software licenses. Relevant aspects are e.g. the types of licenses under which software was made available, and the clauses they contain. *Patent* information describes the owner of a specific patent, or when it was granted.

Large parts of the Context Model of a process can be extracted automatically [17], especially in the aspects of software dependencies and data formats. Other aspects may still require significant manual work to obtain a proper representation. For example, the communication

to a web service has to be described via the provision of its exact address and interface type. Databases usually run as independent server processes they are usually detected but not fully captured when running a tool to monitor a specific research process execution.

We created a set of tools processing eScience Workflows modeled for the Taverna Workflow engine to extract the above-mentioned information and represent it within the context Model. A Taverna2Archi[6] extractor reads Taverna workflow files (t2flow format) and generates ArchiMate models. These are further transformed into OWL ontology representation using the Archi2OWL[7] converter. This way, all information that can be extracted from

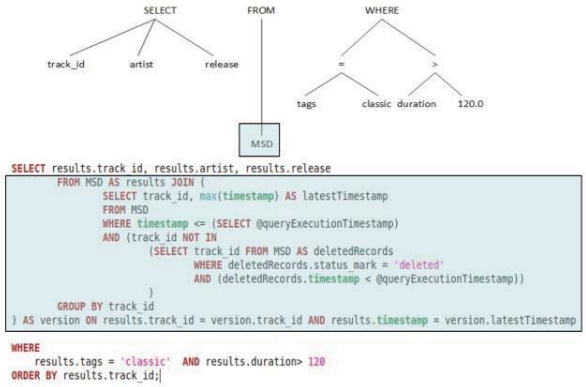


Fig. 2 SQL query selecting data for music classification experiment, supporting data citation

the static workflow definition is captured. This is complemented by monitoring the execution of one or more process execution instances using the extractor of the Process Migration Framework (PMF)[8] which is based on the strace [9] tool. This way, all dependencies are explored and all files and ports touched by the process are detected and added to the context model as dependencies.

These process traces will usually detect an enormous number of libraries and other files used by a process. To refine and make the model more compact, the PMF can resolve Debian packages to which an identified file belongs and therefore create a smaller, concise list of dependencies. It also removes files from the model that are not used for data exchange, for example, log and cache files.

4 Data Citation

Processes frequently process large volumes of data. To be able to repeat any such process we need to ensure that precisely the same sequence of data is fed as input into it. Storing a dump of such huge volumes of data, e.g. as part of the validation data in the context model, is not feasible in big data settings. We need to ensure that we can refer to the original data source / data repository for providing the data upon re-execution. While this may be rather trivial for static data sources being analysed in their entirety, precise identification turns into a challenge when researchers use only a specific subset of the entire data collection, and where this data collection is dynamic, i.e. subject to changes.

Most research datasets are not just static, but highly dynamic in their nature. New data is read from sensors or added from continuous experiments. Additional dynamics arises from the need of correcting errors in the data, removing erroneous data values, or re-calibrating and thus re-computing values at later points in time. Thus, researchers require a mechanism to retrieve a specific state of the data again, to compare the results of previous iterations of an experiment. Freezing the databases at specific points in time, batch-release of versions, etc. all provide rather inconvenient work-arounds, wasting storage space by keeping multiple copies of unchanged data in different releases, and delaying the release of new data by aggregating continuous streams of data into batch releases.

Additionally, most processes will not analyse the entire database, but a very specific subset of it. We thus need to ensure that precisely the same subset can be fed into the process again. Current approaches either waste space by storing explicit dumps of the subset used as input, or require human intervention by providing (sometimes rather ambiguous) natural language descriptions of the subset of data used.

To address this issue, the Working Group on Dynamic Data Citation[10] (WGDC) of the Research Data Alliance (RDA) has devised a set of recommendations to address this challenge. In a nutshell, it relies on time-stamped and versioned storage of the data. Subsets are identified by assigning persistent identifiers (PIDs) to time-stamped queries resolving to the subset. Hash-keys of the queries and the result sets are stored as metadata to allow verification of the resulting data sets upon re-execution [29, 28]. By shifting the focus from citing static data sets towards the citation of queries, which allow retrieving reproducible data sets from versioned data sources on demand, the problem of referencing accurate data sets can be addressed more flexibly. It also provides additional provenance information on the data set by containing a semantic description of the subset in the form of filter parameters in the query. It furthermore allows retrieving the semantically identical data set including all corrections applied to it afterwards by re-executing the timestamped query with a later time-stamp. As the process can be automated it allows integrating data citation capabilities into existing workflows.

[6] ifs.tuwien.ac.at/dp/process/projects/tavernaExtractor.html

[7] ifs.tuwien.ac.at/dp/process/projects/archi2OWL.html

[8] ifs.tuwien.ac.at/dp/process/projects/pmf.html

[9] sourceforge.net/projects/strace

[10] rd-alliance.org/groups/data-citation-wg.html

The persistent identifier serves as a handle which, in addition to representing the input of data in a specific process, can be shared with other peers and be used in publications. As the system is aware of updates and evolving data, researchers have transparent access to specific versions of data in their workflows. There is no need of storing multiple versions of a dataset externally for the long term as the system can reproduce them on demand. As hashing methods are in place, the integrity of the datasets can be verified. Thus the exact data set used during a specified workflow execution can be referenced as part of an experiment description/specification within the parts of the context model describing specific process instances. These can later-on be used for validation.

We implemented several prototypes to demonstrate the feasibility of this data identification and citation approach, including solutions for relational databases (RDBMS) such as MySQL, as well as for comma-separated value files (CSV) [11]. An example demonstrating the query re-writing required to create re-executable queries against time-stamped data is provided in Fig. 2. Audio fulfilling certain requirements (classical music with a minimum length of 120 seconds) is selected as it was available at a given timestamp, removing those that had been deleted by that timestamp.

5 Verification and Validation

Upon re-executing a process (be it a simple reproduction or a repeatability setting after applying preservation actions), we need to verify the correct behavior in a potentially changed environment. To verify and validate the replicated process that was extracted from the source system and run in the target system, we follow the guidelines of [1] that describe the verification and validation of such a *transition activity*. We devised guidelines forming the VFramework [22] which are specifically tailored to processes and describe what conditions must be met and what actions need to be taken to compare the executions of two processes in different environments. This process of verification and validation (V&V) does not check the scientific correctness of the processes. It rather helps in obtaining evidence whether the replicated process has the same characteristics and performs in the same way as the original process.

According to these guidelines, verification checks whether the process set-up and configuration in the new environment is identical to the original one, i.e. whether the same software, operating system, library versions etc. are used in the according configurations. Any changes made to run the process in the new environment (re-compilations, newer versions of individual components) will be detected and reported as potential causes for differences in any reexecution.

Following the static verification, the validation step analyses the actual computations by comparing all interim and final results produced at each input/output point (files, ports) for the original and the re-executed process. This validation data (as well as according metrics) are defined when preparing the VPlan for the process.

The VFramework consists of two sequences of actions. The first is performed in the original environment, i.e. the system that a process is initially deployed in. The results obtained from the execution of each step are written into the VPlan. This VPlan is another modular extension of the context model described in Sec. 3. It contains information needed to validate whether a process is reexecuted correctly. In a nutshell, it comprises of measurement points (usually all input/output happening between the individual process steps), associated metrics (usually testing whether the data for in/output are identical upon re-execution), and according reference process instance data (i.e. storing expected values for specific process test runs to compare against), captured at process runs in the original environment.

The second sequence is performed in the redeployment environment at any time in the future when
the original platform may not be available anymore. The migration of an entire process, i.e. the set-up of a minimal environment required to run the process in an identical configuration, is supported by the second part of the Process Migration Framework. The information needed for such a migration is read from the VPlan. It may, however, be necessary to re-engineer the process to fit it into a new system (in which case the verification step will report all elements in the resulting dependency tree that are different from the original setting).

Subsequently, the validation data is captured again from the re-executed process and compared to the information stored in the VPlan module of the context model using specific metrics. (usually requiring them to be identical or within certain tolerance intervals, depending on the significant properties of the process step/output to be compared.)

We developed the Provenance Extractor [12] which extracts relevant process instance information from the provenance files produced by workflows executed in the Taverna workflow engine. It converts these into an OWL representation linked to the context model via the VPlan module.

We investigated several workflows to define requirements, metrics and measurement points for each of them. The analysis revealed that the majority of functional requirements deal with the correctness of a single workflow step execution and the best way to validate it is to check each of its output ports. In case of the non-functional metrics, the prevailing requirement

[11] datacitation.eu

[12] ifs.tuwien.ac.at/dp/process/projects/ProvenanceExtractor.html

was the computation time that should be similar or should not exceed the 'reasonable time'.

Based on this analysis we validate the workflow by validating all of its steps by comparing the data on the outputs of the workflow steps and also by checking their execution duration. The comparison is made taking into account the format of the data using appropriate tools. For example if two JPEG images depicting the same phenomenon are compared by computing a hash value, they may be detected as being different due to different creation timestamps in the metadata. While this could be fixed by identifying the date of a computation (i.e. the system clock) as one system input being used (which then would need to be set to the same constant value) it may also be modelled explicitly by performing a dedicated comparison for checking the identity of two JPEG files relying only on the image content. A correct way to perform this comparison in general would be to compare the features of the images using software for image analysis. We developed a set of comparator tools for prominent file formats (e.g. HTML, MP3, PDF, PNG, TEX, XML, ZIP) supporting such evaluations. As the Context Model contains information about the file format of data produced/read by a specific step a suitable comparator is selected.

There are a number of challenges that need to be taken into account during V&V. Some processes exchange data with external sources using a variety of network connections. These resources must also be

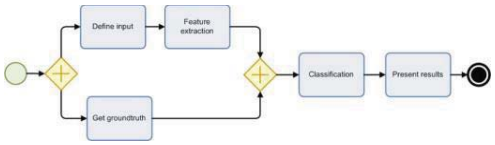


Fig. 3. Music Genre Classification Process [18]

available during the validation process so that the process can interact with them. A solution that allows monitoring of external services for changes, as well as their replacement for the purpose of verification and validation is described in [21].

Another challenge having influence on the verification is the lack of determinism of components. It can apply to both external resources that provide random values and to internal software components that, for example, depend on the system clock or the current CPU speed. In such cases the exact conditions must be re-created in both environments. Potentially, such components need to be substituted with deterministic equivalents [12].

The Context Model contains information about dependencies required to run the software. If any of them was not identified by the automated tools or modelled manually, then the process will not execute. In the course of verification and validation the Context Model gets improved until the process operates correctly. This is achieved either by repeating the capturing of the process using different process instances or by manual addition of identified process dependencies. By verification and validation of the process automatically recreated in the target system we also indirectly verify and validate the Context Model. We determine its correctness and completeness, as the process is re-created via the information stored in the Context Model, re-creating all elements stored there in the target system. If the representation in the Context Model were incomplete the process could not be repeated and run correctly in the target system.

This methodology can be applied to all situations in which a process is re-run, re-produced, or re-used. To support the verification and validation for reproduction and reuse, it is important to also publish the verification data, as other researchers may not have access to the source system. Then they can perform V&V using the validation data provided by the experiment owner.

6 Use Cases

We will use an example from the domain of music information retrieval (MIR) to illustrate the concepts presented in the preceding sections. A common task in MIR is automatic classification of audio into some set of pre-defined categories, e.g. genres such as jazz, pop, rock, classic etc. at different levels of granularities. A process reflecting this task is depicted in Fig. 3. It requires the acquisition of both the actual audio files as well as ground truth information (i.e. pre-assigned genre labels for training and test data in the music collection) from some source. Next, some numeric descriptors (e.g. MFCCS, Rhythm-

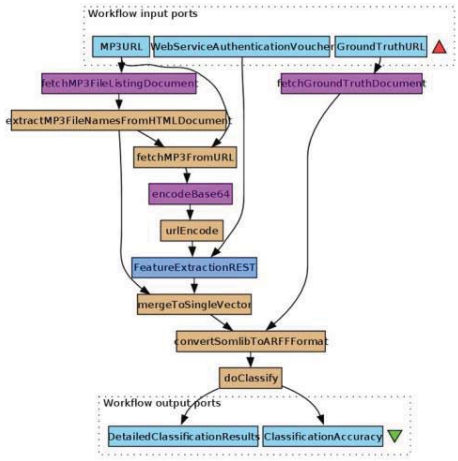


Fig. 4. Music Genre Classification Process modelled in Taverna

Patterns, SSDs) are extracted from the individual audio files via a range of signal processing routines and applying psycho-acoustic models to obtain feature vector representations of the audio. These are subsequently fed into some machine learning algorithm to train a classifier

such as Support Vector Machines (SVM) or Random Forests, and subsequently evaluated using performance measures such as recall and precision.

In one of our experiment settings this process was implemented using a web service for the feature extraction, WEKA as a third-party machine learning package, and a set of dedicated scripts and java applications for tasks such as data acquisition, transformation, etc. These were orchestrated manually via the command line or partially automated via shell scripts, deployed on a Linux system. To increase repeatability and ease automatic analysis we migrated this process into a proper workflow representation using the Taverna workflow engine, as depicted in Fig. 4. It lists explicitly the data sources (URLs) where the audio files and ground truth labels are read from, as well as providing the authentication codes for the web service that the audio files are sent to for feature extraction. The vector files are merged and fed into the classifier which returns the actual classification results and the overall accuracy.

Applying a process monitoring tool we are able to automatically capture all resources (files, ports) accessed or created by one instance of the process, depicted in Fig. 5. This includes, amongst others, a whole range of libraries (depicted in the upper left corner), the set of mp3 audio files (depicted in the lower left corner), a range of processes being called (e.g. wget to download the audio files and ground truth information, depicted in the upper right corner), the user id of the person calling the process, and others.

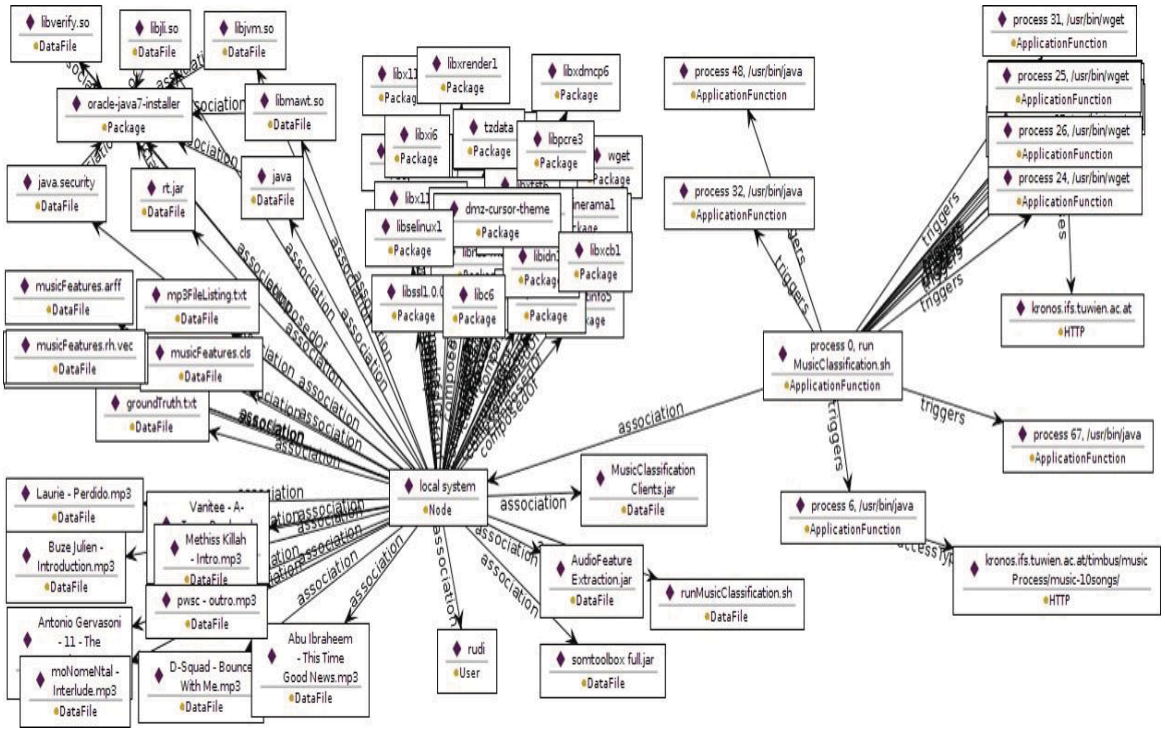
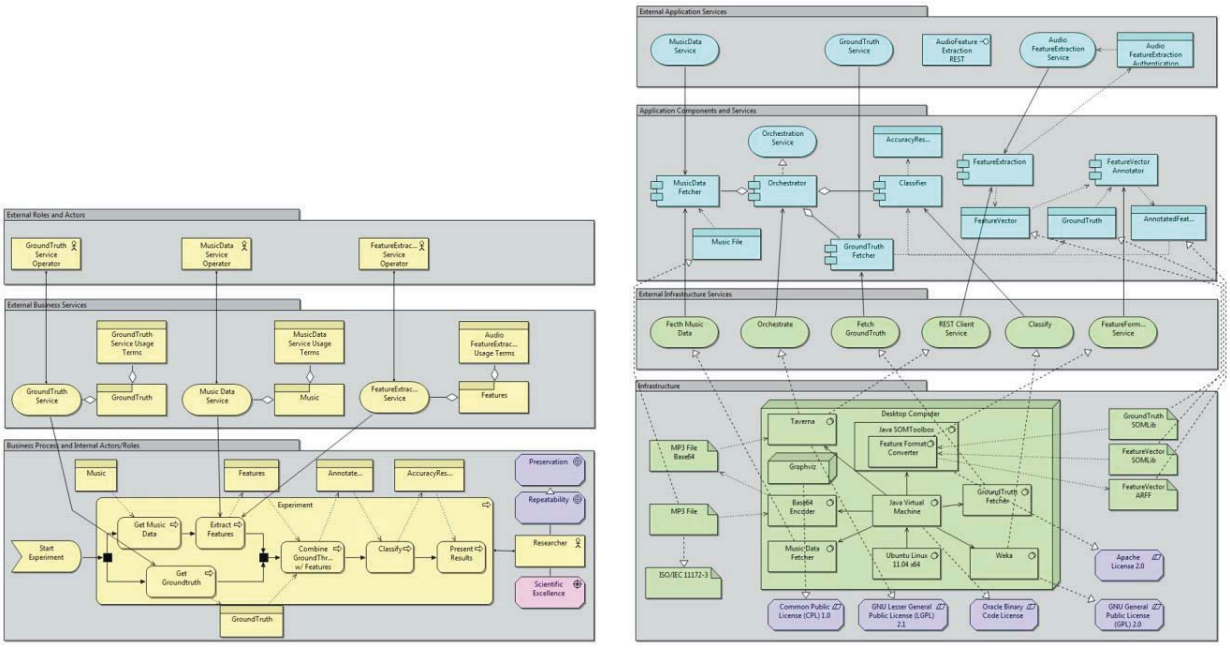


Fig. 5 Dependencies extracted from Music Genre Classification Process.



(a) (b)

Fig. 6 Annotated Context Model of the Music Genre Classification Process.

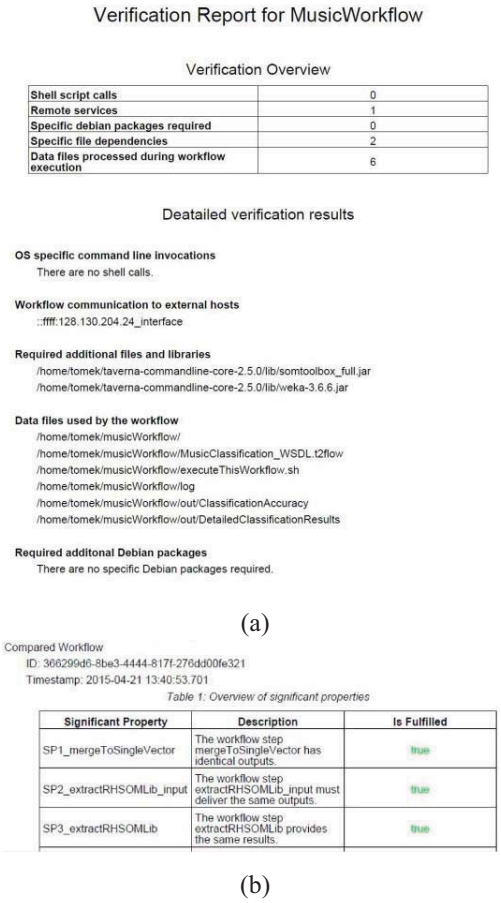
The raw information extracted bottom-up is subsequently enhanced, both automatically as well as manually, by structuring it according to the concepts

The raw information extracted bottom-up is subsequently enhanced, both automatically as well as manually, by structuring it according to the concepts provided by Archimate and adding additional information, such as file format information being added by performing file format analysis using tools such as DROID, contacting file format registries such as PRONOM. The resulting structure is depicted in Fig. 6. Fig. 6a captures, at the bottom, the basic process and the objects (Music files, features extracted and passed on to the classifier, the ground truth annotations, and the final results). Stacked above it are the services being called, i.e. the audio feature extractor. In Fig. 6b the basic software (Java Virtual Machine, WEKA, the data fetchers) are provided, with additional dependencies (e.g. the Unix Bash Shell, Base64 encoders, Ubuntu Linux in a specific version), with the data objects in different representations (e.g. the audio files as MP3 as well as base64-encoded MP3 files) and license information for the various tools (different versions of GPL, Apache License, Oracle Binary Code License, the MP3 patent). On top of these, the detailed application components and services, both internal as well as external, are represented. This way, a comprehensive and well-structured documentation of the process can be obtained in a semi-automatic manner.

This information forms the Process Context Model and can be used for verification and validation.

When applying the VFramework this information is used in two ways. First, to verify the environment in which the workflow is re-executed to confirm that it is configured correctly; second, to validate that the results conform to the original workflow execution. The report summarizing the verification result is provided in fig. 7a. It provides an aggregated summary of the libraries, specifically the jar files of WEKA for machine learning, and the SOMToolbox for the vector format migration, as well as the remote service call for the feature extraction web service.

We use the Process Migration Framework (PMF) tools to generate the VPlan module of the Context Model of a workflow execution and compare it with a Context Model obtained in the same way for its re-execution in a different environment. We use the data captured for each of the workflow steps and compare it using appropriate comparators. For the MIR case study we compare 16 metrics related to the outputs of the workflow steps, thus evaluating 13 functional requirements. We also use 12 metrics related to workflow execution time to evaluate 2 nonfunctional requirements. All of them are fulfilled, therefore the workflow re-execution is established as being repeatable. An excerpt of the validation report is depicted in Fig.7b, confirming that the output at three of the measurement points is identical.



Verification Report for MusicWorkflow

Verification Overview

Shell script calls	0
Remote services	1
Specific debian packages required	0
Specific file dependencies	2
Data files processed during workflow execution	6

Deatailed verification results

OS specific command line invocations
There are no shell calls.

Workflow communication to external hosts
::ffff:128.130.204.24_interface

Required additional files and libraries
/home/tomek/taverna-commandline-core-2.5.0/lib/somtoolbox_full.jar
/home/tomek/taverna-commandline-core-2.5.0/lib/weka-3.6.6.jar

Data files used by the workflow
/home/tomek/musicWorkflow/
/home/tomek/musicWorkflow/MusicClassification_WSDL.t2flow
/home/tomek/musicWorkflow/executeThisWorkflow.sh
/home/tomek/musicWorkflow/log
/home/tomek/musicWorkflow/out/ClassificationAccuracy
/home/tomek/musicWorkflow/out/DetailedClassificationResults

Required additonal Debian packages
There are no specific Debian packages required.

(a)

Compared Workflow
ID: 366299d6-8be3-4444-817f-276dd00fe321
Timestamp: 2015-04-21 13:40:53.701

Table 1: Overview of significant properties

Significant Property	Description	Is Fulfilled
SP1_mergeToSingleVector	The workflow step mergeToSingleVector has identical outputs.	true
SP2_extractRHSOMLib_input	The workflow step extractRHSOMLib_input must deliver the same outputs.	true
SP3_extractRHSOMLib	The workflow step extractRHSOMLib provides the same results.	true

(b)

Fig. 7. (a) Verification and (b) Validation report (excerpt) for the MIR process

13 functional requirements. We also use 12 metrics related to workflow execution time to evaluate 2 nonfunctional requirements. All of them are fulfilled, therefore the workflow re-execution is established as being repeatable. An excerpt of the validation report is depicted in Fig.7b, confirming that the output at three of the measurement points is identical.

7 Conclusions and Future Work

This paper describes a way to move beyond datacentric research evaluation and re-use by addressing the capture and description of entire research processes using Process Management Plans (PMPs), which foster identification, description, sharing and preservation of scientific processes. To demonstrate how the core elements of a PMP can be implemented we described how capturing of computational processes and their context can be performed. We also reviewed the recommendations of the Research Data Alliance on how to precisely identify arbitrary subsets of potentially high-volume and highly dynamic data. Last, we presented mechanisms for verification and validation of process re-executions.

Current work focuses on evaluating the individual components of the PMP with stakeholders from different scientific communities. Specific focus is on tool support to automate the documentation steps, specifically capturing and monitoring of low-level process characteristics and performance aspects. We incorporate all suggestions into a prototype implementation which fosters actionability and enforceability of Process Management Plans.

ACKNOWLEDGMENTS

This research was co-funded by COMET K1, FFG Austrian Research Promotion Agency.

References

[1] IEEE Std 1012 - 2012 IEEE Standard for Software Verification and Validation. Technical report, 2012.

[2] Cristina Aiftimiei, Alberto Aimar, Andrea Ceccanti, Marco Cecchi, Alberto Di Meglio, Florida Estrella, Patrick Fuhrmam, Emidio Giorgio, Balzs Knya, Laurence Field, Jon Kerr Nilsen, Morris Riedel, and John White. Towards next generations of software for distributed infrastructures: The european middleware initiative. In *8th IEEE Intl Conf on E-Science*, 2012.

[3] Australian National Data Service. ANDS Guides Awareness level - Data management planning. Technical Report, 2011.

[4] Paolo Ciccarese, Marco Ocana, Leyla Garcia Castro, Sudeshna Das, and Tim Clark. An open annotation ontology for science on web 3.0. *Journal of Biomedical Semantics*, 2(Suppl 2):S4, 2011.

[5] Andrew Curry. Rescue of old data offers lesson for particle physicists. *Science*, 331(6018):694– 695, 2011.

[6] R. Darby, S. Lambert, B. Matthews, M. Wilson, K. Gitmans, S. Dallmeier-Tiessen, S. Mele, and J. Suhonen. Enabling scientific data sharing and re-use. In *IEEE 8th Intl Conf on E-Science*, 2012.

[7] D. De Roure. Machines, methods and music: On the evolution of e-research. In *2011 Intl Conf on High Performance Computing and Simulation (HPCS)*, pages 8–13, 2011.

[8] David De Roure, Khalid Belhajjame, Paolo Missier, Jos´e Manuel, Rau´l Palma, Jos´e Enrique Ruiz, Kristina Hettne, Marco Roos, Graham Klyne, and Carole Goble. Towards the preservation of scientific workflows. In *8th Intl Conf on Preservation of Digital Objects*, 2011.

[9] Martin Donnelly and Sarah Jones. Checklist for a Data Management Plan. DCC, 2011.

[10] Daniel Garijo and Yolanda Gil. A new approach for publishing workflows: Abstractions, standards, and linked data. In *6th WS on Workflows in support of large-scale science*, 2011.

[11] Ed Gronenschild, Petra Habets, Heidi Jacobs, Ron Mengelers, Nico Rozendaal, Jim van Os, and Machteld Marcelis. The effects of Freesurfer version, workstation type, and macintosh operating system version on anatomical volume and cortical thickness measurements. *PloS one*, 7(6), 2012.

[12] Mark Guttenbrunner and Andreas Rauber. A measurement framework for evaluating emulators for digital preservation. *ACM Transactions on Information Systems (TOIS)*, 30(2), 3 2012.

[13] Tony Hey, Stewart Tansley, and Kristin Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[14] ISO. ISO 5725:1:1994 Accuracy (trueness and precision) of measurement methods and results Part 1: General principles and definitions. Technical report, ISO, December 1994.

[15] K. Belhajjame, O. Corcho, D. Garijo, et. al. Workflow-centric research objects: First class citizens in scholarly discourse. In *Workshop on the Semantic Publishing, 9th Extended Semantic Web Conf*, May 28 2012.

[16] M. Lankhorst. *Enterprise architecture at work*. Springer, 2005.

[17] Rudolf Mayer, Gonc¸alo Antunes, Artur Caetano, Marzieh Bakhshandeh, Andreas Rauber, and Jos´e Borbinha. Using ontologies to capture the semantics of a (business) process for digital preservation. *Intl J. of Digital Libraries (IJDL)*, 15:129–152, April 2015.

[18] Rudolf Mayer and Andreas Rauber. Towards time-resilient mir processes. In *13th Intl Society for Music Information Retrieval Conf (ISMIR)*, 2012.

[19] Rudolf Mayer and Andreas Rauber. A Quantitative Study on the Re-executability of Publicly Shared Scientific Workflows. In *11th IEEE Intl Conf on eScience*, 2015.

[20] Rudolf Mayer, Andreas Rauber, Martin Alexander Neumann, John Thomson, and Gonc¸alo Antunes. Preserving scientific processes from design to publication. In *16th Intl Conf on Theory and Practice of Digital Libraries (TPDL 2012)*. Springer, 2012.

[21] Tomasz Miksa, Rudolf Mayer, and Andreas Rauber. Ensuring sustainability of web services dependent processes. *Intl J. of Computational Science and Engineering*, 10(1/2):70–81, 2015.

[22] Tomasz Miksa, Stefan Proell, Rudolf Mayer, Stephan Strodl, Ricard Vieira, Jose Barateiro, and Andreas Rauber. Framework for verification of preserved and redeployed processess. In *10th Conf on Preservation of Digital Objects (IPRES)*, 2013.

[23] Tomasz Miksa, Stephan Strodl, and Andreas Rauber. Process management plans. *Intl J. of Digital Curation*, 9(1), 2014.

[24] National Science Foundation. Data Management for NSF EHR Directorate. NSF, 2011.

[25] Piotr Nowakowski, Eryk Ciepiela, Daniel Harezlak, Joanna Kocot, Marek Kasztelnik, Tomasz Bartynski, Jan Meizner, Grzegorz Dyk, and Maciej Malawski. The collage authoring environment. *Procedia CS*, 4:608–617, 2011.

[26] Kevin Page, Raul Palma, Piotr Holubowicz, Graham Klyne, Stian Soiland-Reyes, Don Cruickshank, Rafael Gonzalez Cabero, Esteban Garcia, David De Roure Cuesta, and Jun Zhao. From workflows to research objects: an architecture for preserving the semantics of science. In *2nd Intl Workshop on Linked Science*, 2012.

[27] PREMIS Editorial Committee. Premis data dictionary for preservation metadata. Technical report, March 2008.

[28] Stefan Proell and Andreas Rauber. A Scalable Framework for Dynamic Data Citation of Arbitrary Structured Data. In *3rd Intl Conf on Data Management Technologies and Applications (DATA2014)*, Vienna, Austria, August 29-31 2014.

[29] Stefan Pr¨oll and Andreas Rauber. Data Citation in Dynamic, Large Databases: Model and Reference Implementation. In *IEEE Intl Conf on Big Data*, Santa Clara, CA, USA, October 2013.

[30] Van Haren Publishing and A.J.E. Al. *Archimate 2.0: A Pocket Guide*. TOGAF series. Van Haren Publishing, 2012.

[31] D.D. Roure, C. Goble, S. Aleksejevs, S. Bechhofer, J. Bhagat, D. Cruickshank, P. Fisher, N. Kollara, D. Michaelides, P. Missier, D. Newman, M. Ramsden, M. Roos, K. Wolstencroft, E. Zaluska, and Jun Zhao. The evolution of myexperiment. In *IEEE 6th Intl Conf on eScience*, pages 153–160, 2010.

[32] Ralf Treinen and Stefano Zacchiroli. Description of the CUDF Format. Technical report, 2008. http://arxiv.org/abs/0811.3621.

[33] Herbert Van de Sompel and Carl Lagoze. Interoperability for the Discovery, Use, and ReUse of Units of Scholarly Communication. *CTWatch Quarterly*, 3(3), August 2007.

[34] W3C. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Recommendation, 2012.