On the Applicability of Workflow Management Systems for the Preservation of Business Processes

Rudolf Mayer
Secure Business Austria
Vienna, Austria
rmayer@sba-research.at

Stefan Proell
Secure Business Austria
Vienna, Austria
sproell@sba-research.at

Andreas Rauber
Secure Business Austria
Vienna, Austria
arauber@sba-research.at

ABSTRACT

Digital preservation research has increasingly been shifting focus from the preservation of data and static objects to investigate the preservation of complete processes and workflows. Capturing all aspects of a process to be preserved, however, is an extensive and difficult undertaking, as it requires capturing complete software setups with potentially complex setups. Further, the process might use external services that are not easy to capture and monitor. In this paper, we therefore investigate the applicability and usability of using Workflow Management Systems for executing processes in a standard environment where the state of the process can be closely monitored. To this end, we compare three popular workflow management systems. We use a scenario from eScience, implementing a data analysis process, to evaluate the challenges and implications for establishing sustainable and verifyable eScience processes.

General Terms

E-Science, Research Infrastructures, Process Preservation

1. INTRODUCTION

Classical digital preservation has its focus on maintaining the accessibility of digital objects, such as documents, images or other rather static digital files. This area is well understood and solutions to many former problems exist. The next stage of preservation research deals with the preservation of complex systems and composite processes, as they can be often found in the business and research community. Such processes are in contrast to static files highly dynamic and require constant monitoring. The process orientation can be identified in rather young research areas such as E-Science and also the discipline of business process engineering. In science, experiments need to be preserved as researchers need to be able to reproduce and build on top of earlier experiments to verify and expand on the results. It may also proof essential to understand any pre-processing steps and consequences on the interpretation of results in any future meta-studies building on top of earlier research results. In businesses, preservation of processes can play an important role e.g. in liability cases, where a company has to prove that a certain series of steps was executed in the correct manner and according to standards, best practices or laws and regulations. Another motivation are patent litigations, when a company would want to demonstrate how a certain invention originated. Therefore, businesses have to preserve their processes for many years and need to rerun them whenever necessary.

Both areas have in common that they involve large amounts of data and integrate heterogeneous services. These systems form critical infrastructure. Hence their preservation is an urgent and important quest that needs to be tackled. This is a challenging task as these systems are highly complex and consist of many different components, not all of which are under the influence of one controlling instance.

Processes describe how a certain goal has to be achieved. Scientific and business processes consist of intermediate steps which are modelled by the use of workflows. There exist workflow management systems for both domains. These are generic software systems that are driven by explicit process designs to enact and manage operational business or scientific processes, as defined by Aalst[13]. A workflow is defined as “The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.”[13]. Hence, workflows describe the flow of information through a business process. The same paradigm has been adapted in the scientific domain, which lead to Scientific Workflow Management Systems that aid scientists to handle increasingly complex and data driven experiments. In order to tackle this increasing complexity and the orchestration of manifold services and systems, the concept of scientific workflows has received increasing attention within the research community. E-Science projects profit from the combination of automated processing steps in workflows in order to perform complex calculations and data transformations. The advantage of workflows is their capability of adding structure to a series of tasks. They can be visualized as graph representations, where nodes denote processes or tasks and edges denote information or data flows between the tasks. This adds a layer of abstraction and helps to clarify interactions between tasks [2].

Many of today’s data-intensive experiments depend on a number of external service such as Web services, or continuously changing third-party libraries and applications. These changes are not always under the control of the researcher, and may happen at a system level beyond the awareness of the individual researcher, such as e.g. a new library being installed as part of (automatic) system maintenance. This may lead to different results from the workflow, or render the workflow not executable altogether. The possibility to reproduce workflows is also a crucial principle in the business domain.

Preserving the repeatability of such a process in a changing technological environment is therefore a current and emerging topic in Digital Preservation research. Digital preservation of business or E-Science processes requires capturing the whole context of the process, including e.g. dependencies on other computing systems, the data consumed and generated, and more high-level information such as the goals of the process. In this paper, we investigate the feasibility of Workflow Management Systems (WFMS) for preserving scientific processes. We propose that the implementation of a scientific process can be seen as migration strategy, as the original design, structure, meaning and results can be preserved. We provide an overview on the power of such systems and evaluate the effort to migrate workflows between different WFMSs.

The success of preservation activities have to be evaluated. Hence it is required to identify and examine all involved components and the data exchanged between them. This can be achieved by the use of provenance data, which describe the lineage of data and the causal relationships between intermediate steps. Most WFMSs provide the possibility to generate such provenance data automatically. Therefore these systems are valuable for the preservation of processes. In this paper, we first outline which information is important to be captured. We will then investigate the suitability of such automatically recorded provenance data in a case-study of a scientific experiment in the data mining domain.

2. WORKFLOWS AND WORKFLOW MANAGEMENT SYSTEMS

In both domains - science and business - workflows allow to precisely define the involved steps, the required context and the data flow between components. The modelling of workflows can be seen as an abstraction layer, as they describe the computational ecosystem of the software used during a process. Additionally, they provide an execution environment, that integrates the required components for performing a process and executing all defined subtasks. This abstraction supports the preservation process as there is more information about the execution details available. Hence we examine the feasibility of scientific workflow systems for the preservation of scientific processes.

Different scientific workflow management systems (SWMS) exist that allow scientists to combine services and infrastructure for their research. The most prominent examples of such systems are Taverna [9] and Kepler [5]. Vistrails [11] is another workflow management system prominent especially in visualisation, but will not be covered in detail here. Workflow management systems are also prominently used to execute business processes. We will look at the open-source system Activiti.

2.1 Taverna Workbench

Taverna Workbench[1] is an open source project that allows to design and run workflows. It is a general purpose workflow engine that can be used for various applications. It is written in the Java programming language and distributed under the GNU Lesser General Public License (LGPL[2]).

Taverna allows to orchestrate various services and to model the data flow between its components in order to automate a process. Therefore Taverna is widely used in the scientific community and used for modelling data centric experiments. It provides a graphical user interface that allows scientists to design and execute their experiments in a convenient way and to visualize the data flow of an experiment. An example of such a workflow is given in figure 1.

Taverna is a service oriented workflow engine and allows to solve tasks by using either local or remote services. Local services include basic file operations, format conversions and many different tools. Remote services include predefined Web services from various domains, such as bioinformatics or chemistry. It is also possible to implement custom services using the Taverna Java Application Programming Interface (API). Services can also be implemented via scripting language; Taverna to this end supports the language *beanshell*, which is based on the Java programming language.

Taverna uses ports to exchange data between the services: each service can have several input and output ports, where one output port serves as input for a subsequent service. The workbench has an integrated support for scalars and lists, which includes implicit iteration over arrays of data. Looping over data elements is also integrated and allows the usage of control and synchronization points. In its basic configuration, Taverna simply passes down data tokens in a downstream fashion to the next connected service.

2.2 The Kepler Project

The Kepler scientific workflow system[6] is a general purpose application suite for managing, orchestrating and executing scientific workflows. Kepler is an open source project, distributed under BSD license[3] and written in the Java programming language. It provides a graphical interface which enables scientists to design and execute experiments, by linking various services each fulfilling a subtask of a workflow. Kepler inherited the graphical user interface and the actor-centric view of workflows from the Ptolemy II[4] project, which is a framework for designing embedded systems and the communication between components.

Actors are used to model individual steps during the execution of an experiment; they can perform relatively simple tasks as format conversions, displaying data or reading a file from a Web server. There also exist more complex actors that invoke specialized grid services, utilize domain specific databases or execute external services. It is also possible

[1] www.taverna.org.uk/

[2] www.gnu.org/licenses/lgpl.html

[3] http://www.opensource.org/licenses/bsd-license.php

[4] http://ptolemy.eecs.berkeley.edu/ptolemyII/

to develop custom actors by implementing desired features in Java using the Kepler API, and instantiate them within the Kepler system. Further more, python scripts can be executed as well, which reduces the development effort and enhances the flexibility, as no detailed knowledge about the Kepler API is needed.

Actors use Ports for communicating and exchanging data with each other; each Port can either serve as input, output or both to an actor. Ports connect Actors by using channels, which models the data flow and logical sequence of intermediate steps within the workflow. Actors are therefore roughly comparable to services in Taverna.

The workflow is orchestrated by a so-called director, which is the component responsible for arranging the timing of the data flow. There exist different directors for various purposes, such as sequential, dynamic or parallel execution of actors.

2.3 Activiti

Activiti is a workflow and Business Process Management (BPM) Platform, based on the Business Process Modelling Notation (BMPN) 2.0. It is available as open source software and written in the Java programming language, maintained by a consortium of companies offering cloud and Java solutions.

Unlike Taverna or Kepler, it doesn't provide an integrated GUI. Instead, the design of the workflows is enabled by an BPMN 2.0 editor which can be installed as an extension to the Eclipse Integrated development environment (IDE). All the elements available in BMPN 2.0 can thus be used to design the workflow. For execution of the workflow, Activiti can be run as a web-application on a Java Application Server, or as a stand-alone Java application.

Of the BPMN 2.0 elements, most importantly, tasks represent processing steps in the workflow. These tasks are associated via simple sequence flow connections to define the order of execution. Control flow can be modelled with gateways, such as for parallel or exclusive processing. There is no explicit definition of data exchanged, as it is done via Ports in Taverna or Kepler. Rather, a global state of data variables is kept in a key-value map.

Implementation of tasks is enabled by Java classes, or scripting languages that support the Java Scripting Platform, which includes among others JavaScript, Python, Ruby, and Groovy. Both are straight-forward with convenient integration into the BPMN editor. User interaction tasks, which play a more important role in business processes than in scientific experiments, can be implemented via forms; these enable the user to input data, e.g. as workflow input parameters. Unlike the scientific workflow management systems of Taverna and Kepler, Activiti doesn't provide a library of pre-defined tasks to use; however, in the implementation one can draw on the many libraries available to Java and all the script languages supported.

3. CASE STUDY - SCIENTIFIC DATA MINING PROCESS

In this section we discuss the implementation of a typical E-Science process with the workflow management systems introduced above. The specific process used in our case study is a scientific experiment in the domain of data mining, where the researcher performs an automatic classification of music into a set of predefined categories. This type of experiment is a standard scenario in music information retrieval research, and is used with many slight variations in set-up for numerous evaluation settings, ranging from adhoc experiments to benchmark evaluations such as e.g. the MIREX genre classification or artist identification tasks [7].

The experiment involves several steps, which can partially be parallelised. First, music data is acquired from sources such as benchmark repositories or, in more complex settings, online content providers, and in the same time, genre assignments for the pieces of music are obtained from ground truth registries, frequently from websites such as Musicbrainz.org. Tools are employed to extract numerical features describing certain characteristics of the audio files. In the case of the experimental set-up used for the case study, we assume a more complex set-up where an external web service is used to extract such features. This forms the basis for learning a machine learning model using the WEKA machine learning software, which is finally employed to predict genre labels for unknown music. Further, several scripts are used to convert data formats and other similar tasks. The process described above can be seen as prototypical from a range of eScience processes, consisting both of external as well as locally available (intermediate) data, external web services as well as locally installed software used in the processing of the workflow, with several dependencies between the various components.

This scientific experiment has so far been executed by plugging together a number of Java programs, writing their data into intermediate files, and scripts implemented in the Linux shell to provide serial and parallel execution. This set-up does not provide a high degree of resilience to technological changes: the fact that both source data as well as ground truth data are provided externally does not allow the repetition of any experiment with comparable results. Dependencies on software and libraries installed in the experiment platform, that will usually change with frequent system updates further limit repeatability and re-executability. This is further threatened by the fact that the logic of extracting the numeric descriptors is encapsulated in an external service that may update its functional description at any point in time, potentially without providing any information on a service version update.

The scientific process as it is implemented and executed at the moment is exposed to a number of threats. For once, the process is not very well documented, e.g. the exact input and output parameters of each step are not defined, as well as the sequence of execution of the scripts. It is also dependant on the shell of a specific operating system. Even if e.g. the operating system and version is the same, local configuration of the default shell can vary for example on the Linux system, and thus scripts might be not be executable. Monitoring of the process execution is difficult, as there is no direct support available from the shell to capture input and output parameters. Finally, shell scripts might not be

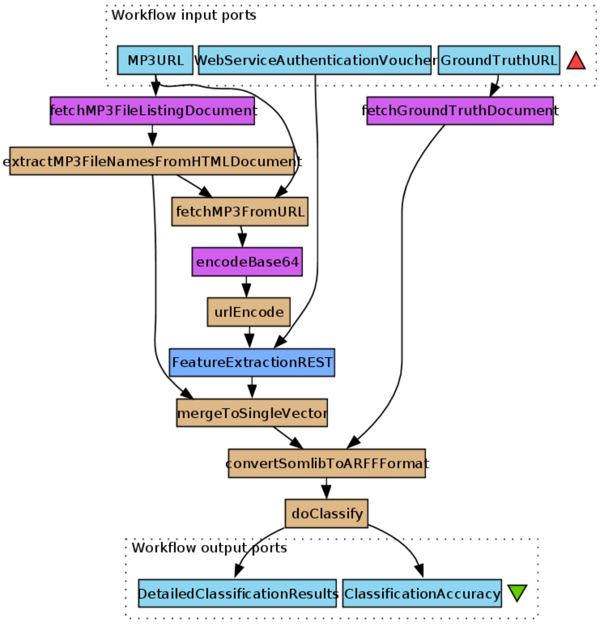

Figure 1: Scientific workflow modelled in the Taverna Workflow engine

persistently stored, but just typed and executed on the shell, and lost upon ending that shell session.

Even though individual of these aspects might be resolved with different means, migration of the process to a workflow management system seems to be a holistic approach towards the digital preservation process.

3.1 Implementation in Taverna

The implementation in Taverna required the migration of scripts and commands, that would have been executed with the shell of the operating system, to scripts in the Taverna-supported language (*beanshell*). These scripts were mainly used for performing format migrations for converting output data into the desired input format for the following tasks. The first step of the workflow could be solved by using services shipped with Taverna. We queried a directory list of a Web server containing the music files to be classified. After the server returned the HTML document containing the URLs of the files, a beanshell script was required to parse the actual locations of the files and for preparing a list. Consequently, the files had to be fetched from the collected URLs. This was achieved by modifying a provided Taverna service slightly to adapt it from images to MP3 files.

The next step (Base64 encoding) could be accomplished with an unmodified Taverna service, but had to be followed by a custom encoding beanshell for ensuring URL safety of the encoded files. The now correctly encoded files are passed to a REST-Web service in the following step. Taverna provides a ready to use REST invocation service, that has to be fed with the according file and an authentication voucher. After the service processed the files, a custom beanshell script was used for merging the single feature vectors to combined file. The resulting file was then converted to the so called ARFF format, by using a beanshell script which invokes a third party library. This Java library had to be provided to the Taverna classpath in advance of the execution, and the usage of the library has to be explicitely specified in the beanshell service using it. After this has been achieved, the API of this library can be addressed via beanshell as if it were regular Java. The final step was again solved by using a beanshell script and an external library, which performs the actual classification.

The implementation in Taverna is fairly straight forward, as it allows to use the power of Java by a simplified scripting language. The library of existing local and remote services is extensive, and these services can easily be adapted to meet required specifications. Another advantage of Taverna is that the design, adaptation and execution of scientific experiments is integrated completely into the workbench, which reduces the installation and administration effort.

3.2 Implementation in Kepler

Kepler also provides scripting capabilities for Python, specifically by Jython[5], an implementation which runs inside the Java Virtual Machine (JVM) and thus does not require additional software packages to be used. Nevertheless, as the third party libraries used in the process are written in Java, we were hinted at implementing the required actors in Java as well. This however is a serious overhead compared to being able to use the third-party library directly from beanshell as in Taverna. To implement the custom actor, one needs to set up the Kepler development environment. The build process is documented well, but requires several steps until the actual development of actors can be achieved.

The workflow implemented in Kepler is depicted in figure 2. A dynamic dataflow (DDF) director is used as there are several loops in the workflow. The first actor activated in the workflow is the Web Service. As invoking the service requires several steps, we encapsulated the internal logic into a so called composite actor. A composite actor itself contains a sub-part of the workflow and is used for reducing the complexity of the overall workflow, by hiding certain parts from the overview.

Although Kepler ships with a large amount of ready to use actors, it was necessary to implement several custom actors for e.g. Base64 and URL encoding on our own. The capabilities of wrapping existing actors, as it is enabled in Taverna, without modifying their source code is limited. Also the implementation of standard dataflow controls such as loops and conditionals requires many small intermediate steps, which render the overall process hard to read, interpret and understand.

3.3 Implementation in Activiti

After setting up the development environment in the Eclipse IDE, the implementation of the workflow in Activiti is rather straightforward, as task definition in the BPMN diagram and implementation of these classes are conveniently integrated. Even though Activiti does not provide a library

[5] www.jython.org/

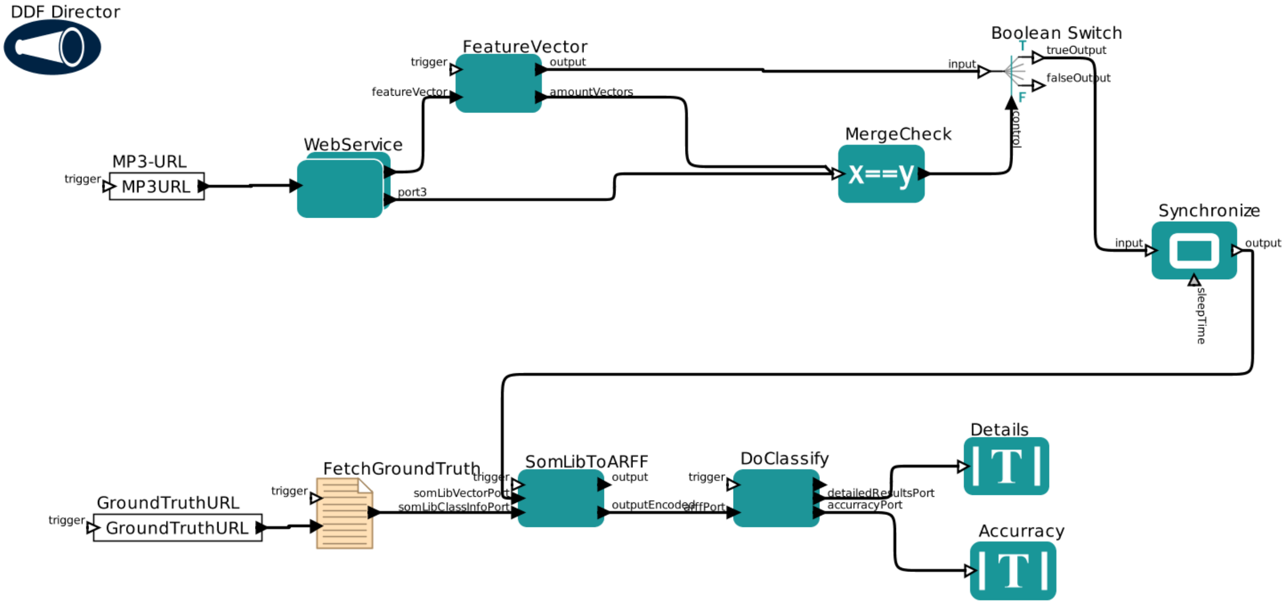

Figure 2: The Music Process Workflow designed in Kepler

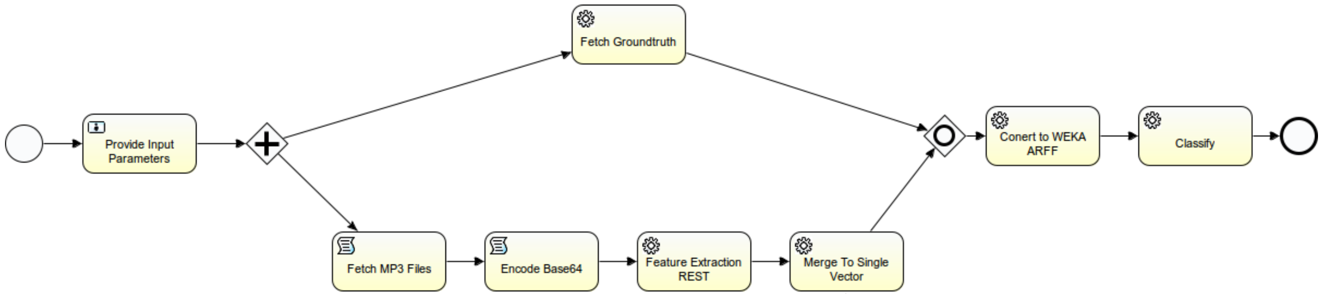

Figure 3: The Music Process Workflow designed in Activiti BMPN 2.0

of commonly repeating task implementations, the straightforward usage of Java as task implementation language allows to draw on the rich set of third-party libraries for compact implementations. Some steps, such as the fetching of files and encoding thereof, have been implemented with in Javascript. Fetching the genre assignment ground truth, calling the feature extraction REST service, converting the data format and performing the classification were solved as Java task implementations.

4. VALIDATION AND VERIFICATION OF PROCESS EXECUTION

Preserving workflows entails the requirement of validating their intermediate results and the overall output of a process execution. Preservation is only then considered a success, if all identified significant properties are equal before and after the preservation. The challenge of keeping workflows accessible is caused by the dynamic nature of processes. External components such as Web services and third party libraries are beyond the influence of workflow designers and users. These components might change at any point in time without prior announcement. Hence, they are critical threats to long term accessibility of workflows. In order to detect these changes, it is necessary to monitor the workflow and the intermediate results they produce. This is a crucial requirement, as otherwise the reproducibility of workflows is at risk.

Measuring changes in significant properties is a difficult task. The authors of [3] propose a framework for evaluating whether two versions of a digital object are identical. The framework consists of several steps, that allow to identify significant properties of digital objects and examine their equivalence after an emulation process. These steps include the description of the original environment and the identification of external resources, that are beyond the influence of a system and influence the object to be preserved. The authors stress that there are different levels, at which objects can be compared with each other. This is also true for workflows. After a workflow has been preserved, the new environment has to be tested if it behaves the same way as the original. Thus test data needs to be used in order to extract and compare the significant properties of the workflow, i.e. if the reaction is identical to the data in both, the original and the preserved environment. The focus of [3] is on emulation as a preservation strategy. The underlying concepts can nevertheless be applied for other preservation strategies as well, that is for instance migrating a workflow, and specifically its components, between different workflow engines.

When preserving processes, the data flow and the causal relationships between involved services can be seen as significant properties. Descriptions of this information is therefore required in order to compare intermediate results with each other. Most workflow management systems use the concept of provenance data to answer questions about execution and design details. Hence provenance data can be used for direct comparison of workflows across workflow engine boundaries.

4.1 Provenance Data

Provenance data describes the lineage of data and provides evidence about execution details, involved services and their intermediate results. A taxonomy of provenance techniques used in various WFMS was introduced in [12]. This taxonomy allows to categorize different systems based on the purpose of recorded provenance, their focus, representation, storage and dissemination. A further distinction between provenance systems can be achieved by the locus of data processing control as identified in [1]. The authors distinguish between command line based data processing, script and program based data processing, query based processing, service based processing and workflow management systems based processing. Depending on the type of data processing, different provenance data can be collected.Provenance data captured during process execution is thus an important aspect that must be captured as process context.

The recorded provenance data can be utilised to verify whether the new version of the process still renders the same results. To this end, as the evaluation framework suggests, one can automatically reapply the inputs and verify the recorded outputs, similar to what would be performed in automated software testing.

The provenance data can further be used for implementing a watch service for software and external service dependencies, e.g. by periodically executing the process with all historic recordings of previous executions, either as a complete process, or for each process step individually.

4.1.1 Provenance Capturing in Taverna

Taverna is capable of capturing the data exchanged between the process steps as provenance data, and stores it in a relational database (Apache Derby). Taverna records all invocations of the workflow and its individual steps, along with the data exchanged and timing information. The data can be exported in the Taverna-specific format Janus [8]; the also available Open Provenance Model format [10] contains only information of the invoked process steps, but not the actual data, and no information about execution time.

An example of the provenance data recorded for the two process outputs, the percentage of correctly classified instances, and the detailed classification results, are given in Listings 1 and 2 (note that some unique identifiers, such as URLs as namespaces, and identifiers for the workflow and specific data elements, have been abbreviated for space reasons).

Listing 1: Example provenance data of Taverna for the process output *ClassificationAccuracy* (cf. Figure 1). The first RDF Description element defines the output port *ClassificationAccuracy*, the second element contains the actual value of “80.0”.

```
<rdf:Description rdf:about=”{nsTaverna}/2010/workflow/{
      idWF}/processor/
        MusicClassificationExperiment/out/
            ClassificationAccuracy”>
  <janus:has_value_binding rdf:resource=”{nsTaverna}/2011/
        data/{idDataGrp}/ref/{idDataPort0}”/>
  <rdfs:comment rdf:datatype=”{nsW3}/2001/XMLSchema#string
        ”>
    ClassificationAccuracy
  </rdfs:comment>
  <janus:is_processor_input rdf:datatype=”{nsW3}/2001/
        XMLSchema#boolean”>
    false
  </janus:is_processor_input>
  <janus:has_port_order rdf:datatype=”{nsW3}/2001/
        XMLSchema#long”>
    0
  </janus:has_port_order>
  <rdf:type rdf:resource=”http://purl.org/net/taverna/
        janus#port”/>
</rdf:Description>

<rdf:Description rdf:about=”{nsTaverna}/2011/data/{
      idDataGrp}/ref/{idDataPort0}”>
  <rdfs:comment rdf:datatype=”{nsW3}/2001/XMLSchema#string
        ”>
    80.0
  </rdfs:comment>
  <janus:has_port_value_order rdf:datatype=”{nsW3}/2001/
        XMLSchema#long”>
    1
  </janus:has_port_value_order>
  <janus:has_iteration rdf:datatype=”{nsW3}/2001/XMLSchema
        #string”>
    []
  </janus:has_iteration>
  <rdf:type rdf:resource=”http://purl.org/net/taverna/
        janus#port_value”/>
</rdf:Description>
```

Listing 2: Example provenance data of Taverna for the process output *DetailedClassificationResults* (cf. Figure 1). The first RDF Description element defines the output port *DetailedClassificationResults*, the second element contains the actual value, one entry for each file tested, with the actual class, the predicted class, and the confidence of the classifier in the prediction.

```
<rdf:Description rdf:about=”{nsTaverna}/2010/workflow/{
      idWF}/processor/
        MusicClassificationExperiment/out/
            DetailedClassificationResults”>
  <janus:has_value_binding rdf:resource=”{nsTaverna}/2011/
        data/{idDataGrp}/ref/{idDataPort1}”/>
  <rdfs:comment rdf:datatype=”{nsW3}/2001/XMLSchema#string
        ”>
    DetailedClassificationResults
  </rdfs:comment>
  <janus:is_processor_input rdf:datatype=”{nsW3}/2001/
        XMLSchema#boolean”>
    false
  </janus:is_processor_input>
  <janus:has_port_order rdf:datatype=”{nsW3}/2001/
        XMLSchema#long”>
    0
  </janus:has_port_order>
  <rdf:type rdf:resource=”http://purl.org/net/taverna/
        janus#port”/>
</rdf:Description>

<rdf:Description rdf:about=”{nsTaverna}/2011/data/{
      idDataGrp}/ref/{idDataPort1}”>
  <rdfs:comment rdf:datatype=”{nsW3}/2001/XMLSchema#string
        ”>
      1   2:Hip-Hop   2:Hip-Hop        0.667 (3.359461)
      2   2:Hip-Hop   2:Hip-Hop        0.667 (3.294687)
      3  1:Classica  1:Classica        0.667 (2.032687)
      4      3:Jazz      3:Jazz        0.667 (2.536849)
      5  1:Classica  1:Classica        0.667 (1.31727)
      6  1:Classica      3:Jazz   +    0.667 (3.46771)
      7      3:Jazz  1:Classica   +    0.333 (2.159764)
      8   2:Hip-Hop   2:Hip-Hop        0.667 (3.127645)
      9      3:Jazz      3:Jazz        0.667 (3.010563)
     10   2:Hip-Hop   2:Hip-Hop        0.667 (4.631316)
</rdfs:comment>
```

Each listing contains two RDF *Description* elements, where the first one defines the output port, and contains as a subelement the identifier of the element containing the actual value, which is the second *Description* element in both listings. With the identifiers used in the *rdf:about* attributes, it is possible to uniquely identify the process step (and iteration, if the step is looped over) the data originates from.

4.1.2 Provenance Capturing in Kepler

The Kepler SWMS provides a dedicated module for recording provenance information[4]. When this software component is loaded, a specialized actor called *Provenance Recorder* is available. This actor is used for monitoring the process execution and storing metadata about the workflow persistently. The provenance module stores provenance data by default in the relationl database HyperSQL (HSQLDB[6]). It is integrated directly into the provenance module and can be queried by using the Database Manager provided by HSQLDB. Kepler stores detailed metadata about every execution of a workflow. This covers actors, ports, parameters, relations and additional information about the workflow, such as user names and context information. The Kepler provenance system also stores the data used during the execution, which allows the detection of changes within the results.

All information stored within HSQLDB can be queried by using standard SQL, and from a Java program via an API. The OPM export feature completes the provenance data management of Kepler; in contrast to Taverna the exported OPM XML file contains time stamps and allows to derive the execution sequence easily.

Listing 3: A Kepler OPM XML snippet

```
<wasGeneratedBy>
    <effect id="_a2"/>
    <role value="output"/>
    <cause id="_p0"/>
    <time>
        <noLaterThan >16:26:17.333+02:00</noLaterThan>
        <noEarlierThan >16:26:17.333+02:00</noEarlierThan>
        <clockId>_c1</clockId>
    </time>
</wasGeneratedBy>
```

Listing 3 depicts an example of an exported OPM file. It contains references to the actor that generated the output (_p0) and refers to the the output of this event (_a2), using auto-generated identifiers to refer to these elements.

4.1.3 Provenance Capturing in Activiti

Activiti refers to provenance data as (process execution) history, and allows to configure recording on several levels of detail. Similar to the other systems, the data is stored in a relational database (H2[7]), which can be queried to retrieve information about the process and task invocations. As there is no explicit input and output of process steps (ports in Taverna and Kepler), rather the global state of data in the process execution is stored, than specific parameters for a specific task invocation. Activiti also does not provide an export into e.g. the OPM format.

5. COMPARISON OF WORKFLOW MANAGEMENT SYSTEMS

We identified a number of criteria important for the migration and execution of processes workflow management. A summary of these criteria is provided in Table 1.

Regarding setup of the design and execution platform, Kepler and Taverna provide a straightforward installation routine, while Activiti requires a bit more work with preparing the Eclipse IDE and plugins.

All systems evaluated in this paper allow to implement the process with the use of the Java programming language, even though the complexity of doing so differs greatly; both Kepler and Taverna require the programmers to develop the modules outside the workflow management system and then to register their services and actors, respectively, with the engine. Implementing tasks in Activiti benefits from the initial setup of the Eclipse environment.

The systems differ greatly when it comes to the support of scripting languages for fast and simple tasks. Here, Activiti provides the widest range of languages, and is in theory not limited, as long as the language conforms with the Java Scripting Platform. If there are a lot of legacy scripts that would need to be preserved, Activiti would thus seem to be a prime choice. It seems vital that other systems would allow such a wide range of implementations as well. Still, this will also raise the complexity of preserving the actual process as components in many different languages may need to be preserved, together with their operational infrastructure (compiler, interpreter, runtime environments, etc.). Kepler provides Python, and Taverna Beanshell scripting capabilities. The latter further provides a large library of services that can be used to quickly perform common tasks, and allows to easily alter these template implementations.

All systems allow to record provenance data during the process execution, which enables for validation. Kepler and Taverna provide various types of exports of this data, in the Open Provenance Model (OPM) and custom formats, and are more detailed on the single processing steps, as input and output ports of each process step are clearly defined. This seems to be an important aspect for detailed validation and watch activities. Activiti could be easily augmented by an export into the OPM, and input and output parameters for a processing step could, for a specific process execution, be deduced from the change in global variables.

6. CONCLUSIONS

The preservation of complete (business) processes is starting to be understood as a new challenge in digital preservation research. Scientific processes need to be preserved to allow later verification of results, and business process preservation can play an important role when it comes to liability or patent infringement litigations.

However, process preservation is inherently difficult – today's processes are executed inside complex software ecosystems, and composed of a myriad of services. Capturing this software setup and its configuration is only one step – without being able to validate the process execution, we cannot

[6] www.hsqldb.org/

[7] http://www.h2database.com

Table 1: Features of Workflow Management Systems

Engine	Implementation	Script Language Support	Designer Support	Execution Engine	Provenance Capturing	Provenance Export
Taverna	Java	Beanshell (Java)	Standalone	Integrated with designer	Database (Apache Derby)	OPM & Janus
Kepler	Java	Python	Standalone	Integrated with designer	Database (HSQLDB)	OPM
Activiti	Java	JavaScript, Python, Ruby, Groovy, ...	Via Eclipse IDE	Web application or Java program	Database (H2 DB)	-

guarantee that the preserved process is still the same when re-executed at a later time.

These two concerns are a bit relaxed when defining and executing the process in a dedicated workflow engine, which provides a layer of abstraction to the original software setup. It also allows to closely monitor, and thus evaluate, the process execution. In this paper, we therefore described a number of popular workflow management systems, and described how they can be used to migrate a scientific experiment process. Efforts for migrating a workflow to a workflow management system might be significant; therefore, flexibility in the implementation is a prime aspect.

With the migration of a process to a workflow management engine, we can mitigate a few concerns that can hamper the preservation of this process. First, the migration to workflow engines has the benefit of requiring a clear and formal definition of the processes, which might not be present before. Thus, we obtain documentation and detailed descriptions on the process. Further, we can evaluate and monitor the execution of the processes closely, which enables verification that a process is still executed unchanged. Finally, the migration to a workflow management system in general is a step of abstraction from a specific software setup. The requirements and interfaces to operating systems or and system libraries are greatly reduced, and dependencies on third-party libraries are generally explicitly defined.

The migration does not prevent external elements such as the webservice employed in our case study from becoming obsolete. Thus, contracts and service level agreements have to be agreed on with the providers of these services to maintain and migrate their services if needed. Then, using previously recorded provenance data, we can verify whether these services still behave as before.

7. ACKNOWLEDGMENTS

Part of this work was supported by the projects APARSEN and TIMBUS, partially funded by the EU under the FP7 contracts 269977 and 269940.

8. REFERENCES

[1] R. Bose and J. Frew. Lineage retrieval for scientific data processing: a survey. *ACM Comput. Surv.*, 37(1):1–28, Mar. 2005.

[2] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science and Engg.*, 10(3):11–21, May 2008.

[3] M. Guttenbrunner and A. Rauber. A Measurement Framework for Evaluating Emulators for Digital Preservation. *ACM Transactions on Information Systems (TOIS)*, 30(2), 2012.

[4] The Kepler Project. *Getting Started with Kepler Provenance 2.3*, August 2011.

[5] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[6] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, Aug. 2006.

[7] Music Information Retrieval Evaluation eXchange (MIREX). Website. `http://www.music-ir.org/mirex`.

[8] P. Missier, S. S. Sahoo, J. Zhao, C. A. Goble, and A. P. Sheth. *Janus*: From workflows to semantic provenance and linked open data. In *Proceedings of the International Provenance and Annotation Workshop (IPAW2010)*, pages 129–141, Troy, New York, USA, June 15–16 2010.

[9] P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble. Taverna, reloaded. In M. Gertz, T. Hey, and B. Ludaescher, editors, *SSDBM 2010*, Heidelberg, Germany, June 2010.

[10] L. Moreau, J. Freire, J. Futrelle, R. E. Mcgrath, J. Myers, and P. Paulson. *Provenance and Annotation of Data and Processes*, chapter The Open Provenance Model: An Overview, pages 323–326. Springer-Verlag, Berlin, Heidelberg, 2008.

[11] C. Silva, J. Freire, and S. Callahan. Provenance for visualizations: Reproducibility and beyond. *Computing in Science Engineering*, 9(5):82 –89, sept.-oct. 2007.

[12] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, Sept. 2005.

[13] A. van der Aalst, A. Hofstede, and M. Weske. Business process management: A survey. In M. Weske, editor, *Business Process Management*, volume 2678 of *Lecture Notes in Computer Science*, pages 1019–1019. Springer Berlin / Heidelberg, 2003. 10.1007/3-540-44895-0_1.