Evaluation of Preserved Scientific Processes

Rudolf Mayer[1], Mark Guttenbrunner[1], and Andreas Rauber[1,2]

[1] Secure Buisness Austria, Vienna, Austria
[2] Vienna University of Technology, Austria

Abstract. Digital preservation research has seen an increased focus is on objects that are non-deterministic but depend on external events like user input or data from external sources. Among those is the preservation of scientific processes, aiming at reuse of research outputs. Ensuring that the preserved object is equivalent to the original is a key concern, and is traditionally measured by comparing *significant properties* of the objects. We adapt a framework for comparing emulated versions of a digital object to measure equivalence also in processes.

1 Introduction

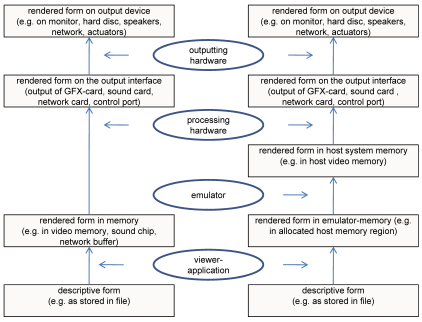
Digital data is in its nature volatile, and always needs an environment in which it is rendered to a form that makes it useful. The interpretation of the bitstream is the subject of digital preservation (DP) research. Two strategies have evolved as the most promising. *Migrating* a digital object means continuously changing its format to one that is not obsolete at the time of use. *Emulation* keeps the original digital object but changes the rendering environment, by modifying the application used to render the object, or replacing the original hardware by introducing a virtual layer, using the original software-stack for rendering. It is necessary to evaluate that the result of the digital preservation action produces a rendering that is similar to the original in its "significant properties", i.e., the properties of the object deemed important for future use by the designated community. For migration, the significant properties of the object extracted from the original and migrated form are compared. With emulation, one has to compare the rendering of the digital object in the two environments. Traditionally DP research concentrated on objects that behave deterministic, i.e., are rendered similarly on the same system during each rendering, such as text-documents, videos, images, or database content. Objects that are rendered non-deterministic change their rendering depending on user input, hardware values, or random values, e.g. digital art or computer games. The preservation and curation of scientific data is important to ensure reuse and long-term usability of data that has been the basis to scientific experiments. Data, however, does also require information about its context and the processes involved in its creation and usage, e.g., the setting where it was created or interpreted. It may be impossible to recreate the original experiment, and thus a preserved process that allows to reproduce and trace how results and analysis were obtained is important. eScience processes are depending on specific software or hardware, thus facing digital obsolescence. To

preserve a process, one needs to go beyond capturing single files and metadata, up to including complete computer systems. From a high level perspective, organisational parameters need to be described, down to the technical description of the systems the process depends on, including hardware, software, and third-party services. To describe these, a context model identifying relevant aspects of a process has been created [2].

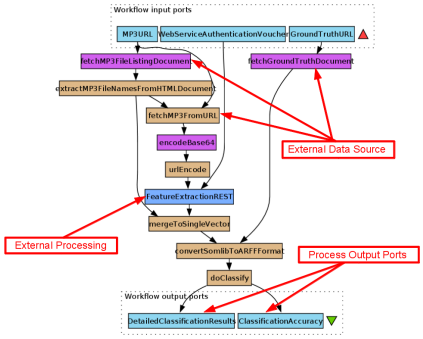
2 Preservation Actions for Processes and Evaluation

The preservation of a process will in most cases be a mixture of currently available techniques. Documentation relevant to the understanding of the process and technical environment can be migrated to different formats. Also data and documents used within or in the transition between process steps need to be preserved. They can both be interpreted by humans, or machines, which has an impact on the significant properties. For software systems forming the execution environment supporting the process, emulation of both hardware and software (especially operating systems) is a viable option. Virtualisation can aid to abstract the system from the physical hardware. Migration of the software supporting the process to another environment is a viable option, e.g. by migration to a different programming language supported in the new environment, or cross-compilation to a different platform. One strategy for external systems employed, such as web services, is in contractual agreements, which will obligate the providers to perform preservation efforts themselves. When that is not possible, an external system has to be replaced by a system controlled by the process owner. This can be a re-implementation (migration) of the system, or simply a simulation of the behaviour of the system, e.g. by recording and replaying messages previously exchanged. Of course, this is not a valid strategy for non-deterministic services, e.g. for which the output depends also on the time of invocation (and thus a *state*). In all cases, it is important to evaluate that the preserved process is still equivalent to the originally executed process.

A framework to determine the effects of an emulated environment on the rendering of objects is presented in [1]. It suggests methods to automate the evaluation, and methods to automate input to ensure that changes in manual handling of the digital object can be ruled out as a cause for changes. To apply this framework to the evaluation of processes we have to take the following steps:
(1) *Description of the original environment*, using the context model.
(2) *External events* Typical events that will influence the behaviour of a process are external data used as input to the process, either manually supplied by a user, or data that is read from sources connected to the system, e.g., web services, or sensors. To enable evaluation, we have to record the data supplied to the process so that it can be reapplied on a re-run. As not every possible combination of data can be evaluated, significant test cases are defined.
(3) *Level to compare* Depending on the process and the type of "output" it creates, we have to decide where to capture the rendered data (cf. Figure 1(a)). For an automatically running process that does not produce any rendered screens



(a) Forms of a digital object in the original and emulated system's memory



(b) Scientific workflow modeled in the Taverna Workflow engine

for users we can compare on the interface where the output is provided, e.g. a file. If user output is created and that triggers a response from the user, then extraction from the video memory e.g. in the form of a screenshot is necessary.
(4) *Recreating the environment* A typical recreation of a process would happen either in a virtualised or a fully emulated environment. Alternatively the process could also be executed in a different process execution engine.
(5) *Standardised input* External data that has been captured during the test case-recordings of the rendering of the process has to be provided to the new rendering of the process, to ensure a deterministic rendering.
(6) *Extracting data* During the execution of a process intermediary data is generated, handed between different steps of the process and discarded afterwards. From an evaluation point of view we are interested in all data that is provided outside of the system. We thus identify these "Significant States" in the process and at these extract both in the original rendering and the emulated version. If the original process is executed in a workflow engine, i.e. an application that orchestrates processes, the single processing steps are well defined and allow for capturing the data exchanged. For other processes, a viable strategy is to move the execution environment into a virtual machine environment, which enables capturing data from the interface between the virtual machine and the host system. Capturing data in the original environment is also possible to some extent.
(7) *Comparing data* The data extracted during the run of the process in the original environment is compared to the data extracted from the re-run. Using the same input data for a process the assumption is that the process behaves similar to the original rendering, producing the same results if we extract the data in the same significant states of the process. Otherwise, a difference in the rendering environment is likely the reason for the failed verification of the process.

3 Case Study – A Music Classification Experiment

We test our method on a scientific experiment where the researcher performs an evaluation of an algorithm for automatic classification of music into a set of

predefined categories, which is a standard scenario in music information retrieval research, and is used for numerous evaluation settings, ranging from ad-hoc experiments to benchmark evaluations. An implementation of the process is shown in Figure 1(b), a detailed description and context model thereof can be found in [2]. In the experiment, music data is acquired from external sources (e.g. online content providers), and genre assignments for the pieces of music from ground truth registries. Features describing characteristics of the audio files are extracted using an external web service. The service needs the input in MP3 format, Base64 encoded to allow for a data exchange via XML. The service returns an ASCII file, that after conversion from SOMLib to ARFF format forms the basis for learning a machine learning model. The numeric metric "accuracy" and a detailed description are the output.

Of particular interest for evaluation are those steps where the process communicates with the system it is embedded in, i.e., all the steps where external data influences the rendering of the process, or where data is provided to external interfaces. To create a deterministic rendering we need to make sure that for every evaluation cycle the same data influencing the process is provided. In the music classification process there are several process steps that have a connection to an external service, annotated in Figure 1(b). As such, fetchMP3FileListing, fetchGroundTruth and fetchMP3 get data from an external web server. Communication with the web service includes the music files as well as parameters controlling the extraction. The process in turn receives as input the extracted features. Finally, there are two end results where data is provided by the process.

To enable evaluation of the workflow results we need to provide either a connection to the original external web server, or create a simulation of the service. Capturing the data on the output ports of the process lets us then compare different cycles of the same process steps with the same data. If the captured data is identical even if the rendering environment has been changed, we have a strong indication that the process is executed correctly in the new environment. If the captured data is different, even if the process has been executed with the exact same input parameters, then the rendering environment changes the execution of the process and thus can not be considered a proper preservation of the process. Also internal intermediary results after each step can be compared, to check how far in the process the results are unchanged. Differences can be introduced by internal dependencies, e.g., due to differences in versions of libraries, or different versions of the Java Virtual Machine. Comparing intermediary results can thus help us identify differences in the rendering environment leading to changes in the result of the process.

References

1. M. Guttenbrunner and A. Rauber. A measurement framework for evaluating emulators for digital preservation. *ACM Trans. on Information Systems*, 30(2), 2012.
2. R. Mayer, A. Rauber, M. A. Neumann, J. Thomson, and G. Antunes. Preserving scientific processes from design to publication. In *Proc. of the 15th Int. Conf. on Theory and Practice of Digital Libraries*, Cyprus, September 23–29 2012. Springer.