Risk Driven Selection of Preservation Activities for Increasing Sustainability of Open Source Systems and Workflows

Tomasz Miksa, Rudolf Mayer
Stephan Strodl, Andreas Rauber
SBA Research, Vienna, Austria

Ricardo Vieira, Goncalo Antunes
INESC-ID Information Systems Group
Lisbon, Portugal

ABSTRACT

The increasing demands faced by repository systems and the growing popularity of workflow systems introduces new risks, creating new challenges to the digital preservation community. The application of risk management practices to digital preservation is a way of managing the risks associated with the use of such systems and to optimize the application of digital preservation treatments to such risks. In this paper, we present results of a case study conducted on two use cases: a repository system and an automated workflow representing a typical digital preservation quality assessment process. We used a risk assessment approach to identify risks related not only to the technical but also organizational and legal aspects. We assigned controls which decrease their impact and explained how the digital preservation related controls can also improve the current functioning of the repository system and increase reproducibility of the workflows.

1. INTRODUCTION

In recent years, the digital preservation community has been investigating different ways of dealing with the preservation of static contents like scans of books or music recordings. Several solutions were proposed and successfully implemented. They range from frameworks and metadata vocabularies to distributed repository systems. Keeping up with the paradigm shift in science and the deluge of data [17], the digital preservation solutions are being enhanced to address the requirements of preserving complex objects like scientific data, workflows and processes. Nowadays, the digital preservation actions aim not only to safeguard the heritage to future generations but also to enhance the reproducibility of data-driven research.

There are several studies on repository systems which compare their functions and evaluate whether they address the needs of institutions in need of such [12]. Most of the systems are open source, which can be related to the fact that it is often assumed that open source solutions are considered to have higher preservability than their commercial counterparts. Such an assumption is also taken for the reproducibility of modern research, i.e. the scientific experiment is supposed to be reproducible when it is published under an open source license.

Yet, this assumption can be challenged. Being open source is not, by itself, a guarantee of the higher longevity of repository systems. At some point their preservation will be required, which can be a challenge due to the fact that these systems are more than databases which collect metadata and store the preserved objects. These systems are quite often distributed, benefiting from the integration of several external services which are provided by external entities. There is a potential risk that the functionality of the system can be severely affected when one of these services becomes unavailable or changes are made to its functionality.

In the worst case this may hinder the possibility of retrieving and presenting the preserved objects to the user. Similar threats are also affecting scientific workflows, which very often contain references to external services. Furthermore, workflows often require access to external libraries and software applications which need to be present during execution but are not explicitly defined in the workflow specification.

For this reason, we investigated potential threats to open source repository systems and workflows. A case study was conducted on two use cases: a repository system based on Fedora Commons, and a typical Taverna workflow used during preservation quality assessment. Both of them are available under the open source licenses. We used a risk assessment method based on ISO 31000:2009 and aligned with the TIMBUS preservation framework. The case study identified a wide spectrum of risks related not only to the technical but also organizational and legal aspects. We assigned controls which help to decrease their impact and detail the solutions delivered by the TIMBUS project that help to control the risks which are related, not only to digital preservation, but also to the current functioning of the repository system and reproducibility of modern research.

The paper is structured as follows. Section 2 presents the state of the art in risk management and explains the process preservation framework which guided our assessment. Moreover, changes in the web services that may affect both the

repository system and the scientific workflows are discussed in this section. Section 3 describes two use cases on which the case study was performed. In Section 4, the approach used for risk assessment of both cases is presented and the results are discussed. Section 5 describes selected controls applied to both use cases. Finally, conclusions are presented in Section 6.

2. STATE OF THE ART

This section explains the risk management approach applied on the case study and provides an overview of available standards. It also places the risk management process within the process preservation framework. Finally, possible kinds of changes in web services are discussed.

2.1 Risk Management

Risk Management (RM) concerns the assessment and control of risks, with risk being defined as the combination of the likelihood of an event and its consequences [9]. Its ultimate goal is to manage the uncertainty associated with risks, either by mitigating risks with negative consequence on objectives or by taking advantage of risks with positive consequence on objectives [9].

Although different standards, methods and tools exist for targeting specific domains, ISO 31000:2009 [11] describes a generic and domain-independent framework for risk management, providing the underlying concepts and principles, along with a process. The risk management process defined by the standard is depicted in Figure 1.

The process starts by defining the internal and external context of the project. The external context might consist of a description of the regulatory environment of the project or any other element that might affect data management. The internal context includes defining all the elements of the project, i.e. its objectives, resources, data, processes, systems, among others that may be relevant to consider.

After establishing the context, the assessment of the risks based on the collected information is performed. It is composed of three different steps: (1) risk identification, where all relevant assets, vulnerabilities, events and risks are identified; (2) risk analysis, where the value of the assets, the exposure to vulnerabilities, the likelihood of events, the risk consequence, and ultimately the risk severity are estimated; and (3) risk evaluation, where the information produced in the two previous steps to check against risk criteria is evaluated, culminating in a decision on whether a specific risk is acceptable or tolerable. Depending on the context of the risk assessment, different risk assessment techniques can be applied to the process. The standard describes several of those techniques and their suitability for the different steps of the process.

These risk assessment steps result in the prioritization for risk treatment, with the identification of controls. If the controls are sufficient to lower the overall risk level into acceptable values, then a risk report is defined. All the steps of the process should be communicated to the interested parties for consultation and validation. Additionally, the process should be run continuously, with constant monitoring and review of the different steps, if necessary, so that

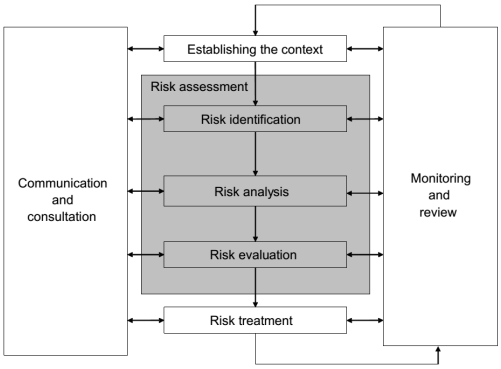

Figure 1: Risk management process according to ISO 31000:2009 [11]

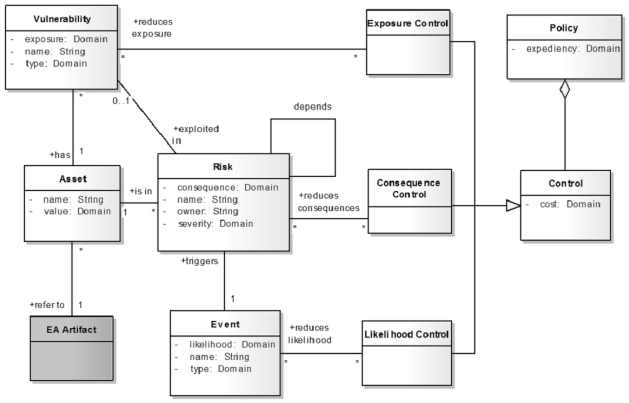

Figure 2: Risk concepts [2]

the risk management is effective.

Digital preservation is one of the domains where risk management has been applied, as it is about recognizing that during its lifecycle, data is subject to risks that can affect their proper use and interpretation. Different works concerning risk management applied in this domain have been published, including the ISO 16363:2012 [10], that provides a risk management process for assessing the trustworthiness of digital repositories, and the Digital Repository Audit Method Based on Risk Assessment (DRAMBORA) [13], which also describes a process for assessing digital preservation repositories. Additionally, in [3], the authors identify a set of typical threats and vulnerabilities that can be mitigated using Digital Preservation techniques.

TIMBUS proposes a risk management-based approach to the preservation of business processes [15]. In that sense, digital preservation is seen as a risk treatment, with the clear interfaces existing between the TIMBUS processes and the risk management process. The risk management process adopted by TIMBUS follows the ISO 31000:2009 standard. A conceptual model is used along with the process, defining a set of risk management concepts, based on the work described in [2]. The model, which can be seen in Figure 2, is based on the ISO 31000:2009 family of standards, and was created to support sharing, reuse and processing of risk concepts. The model defines risk as "an effect of uncertainty

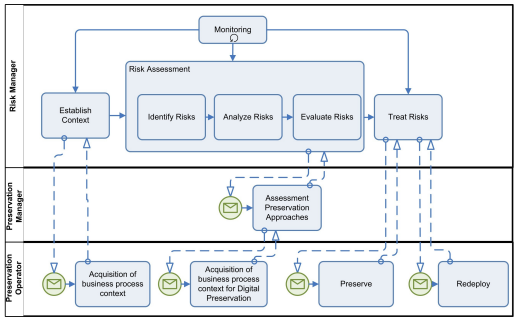

Figure 3: TIMBUS framework for process preservation, BPMN model

and is expressed by the combination of the likelihood of an event and its consequences when exploiting a vulnerability of an asset" [2]. Asset is defined as something (e.g. process, data, hardware, software, people) that has value to the project. A risk is expressed by a risk severity (or risk level) that is a combination of its consequence with the likelihood of the event triggering the risk. Finally controls are defined as actions that can be taken to mitigate risks. Controls can reduce the exposure of a vulnerability, reduce the likelihood of an event, reduce the risk consequence, transfer the risk and accept the risk. A risk policy represents a set of controls that were applied to mitigate the risks in a specific context.

2.2 Framework for Process Preservation

A process model for digitally preserving a process is described in detail in [16], and depicted in Figure 3. It is centred around the risk management approach detailed above, and can be divided into three phases: *plan*, *preserve* and *redeploy*.

The plan phase concerns the capture of the business process context. To this end, a context meta-model is used to systematically capture these aspects that are essential for its preservation and verification upon later re-execution [1]. This model is implemented in the form of an OWL ontology, which provides both generic core concepts and domain-specific concepts that are used for capturing information in specific domains. The generic concepts are based on ArchiMate [7], which provides a template to describe a business by around 30 different concepts on the business, application and technology layer. A number of domain-specific concepts dealing with *Software licenses* and *Patents*, *Software application dependencies*, and *Digital preservation meta-data*, among others, are provided.

Assessment of Preservation Approaches is responsible for the identification and evaluation of different preservation approaches (controls) for the process. Some specific controls will be discussed together with the use case description in Sections 4 and 5. Each approach is specified in a *Process Preservation Plan*, which also defines procedures for capturing the process data and later redeploying and verifying the process.

During the preservation phase, these controls are applied, and validation and verification data are captured from the source system for redeployment. The redeployment phase specifies the re-initiating of the preserved process in a new environment at some point in the future. Necessary adjustments to the target environment are performed, and finally, the process can be re-executed and the taken measurements can be validated.

2.3 Changes in services

According to the classification presented in [14], there are four ways in which a web service may change. In this section, those changes are discussed, as they may apply to these use cases and, additionally, because the general classification may apply to any kind of service (not only web services).

A web service can become unavailable. This will likely stop execution of the process, unless alternative paths and exception handling has been implemented for such a case. Reasons for unavailability can range from temporary technical problems, to bankruptcy of the service provider. Such situations are straight forward to detect, for instance, by using time-outs which would alert to the unavailability of the web service.

A web service can change its communication interface, not always jeopardising the full execution of the process. Such a situation may also be easily detected. It may require short pauses in the process execution until the changes are adopted into the process. Of course, in case of significant changes in the communication interface (e.g., switch from REST to WSDL), time needed for reconnecting the web service into the process may require more effort.

The functionality of a web service may change, which denotes that the outputs of the web service change, while the interface stays the same. Unlike the first two threats, this threat is hard to detect, as the process may not break, but instead will be delivering outputs which are not correct or are different from expected. Such a situation might be detected only much later and on a different level, e.g., when some general statistics regarding process performance are changed. Such a situation may occur for several reasons. One of the reasons may be the changes at the semantic level, e.g., switching the unit of measurement from inches to centimetre due to a server configuration change. Other possibilities are bug fixes in the underlying algorithm (which may introduce other bugs as well), or intentional changes in the functionality, e.g., faster but less accurate computational algorithms.

A web service may change its non-functional behaviour, which may not always stop the process from correct execution, but can occur temporarily and therefore be hard to notice. The examples of such cases could be different timing characteristics or delays, effects of buffering, etc. They also need to be detected, because there may be a threshold from which the web service cannot deliver its functionality properly, and therefore stop or alter execution of a dependent system.

3. USE CASES

In this section, an overview of two open source use cases is presented. The technical aspects that are relevant for the risk assessment presented later in this paper are explained. The first use case deals with a repository system, while the second use case is a typical digital preservation workflow.

Different as they may seem, the later analysis shows that they have much in common. We used these two cases in order to demonstrate the broad applicability of the TIMBUS preservation framework and the risk driven approach.

3.1 Fedora Commons Based Repository

This use case concerns a real installation of a repository system at a university. Due to the fact that the analysis described in this paper may reveal sensitive data, we cannot disclose any information which would allow its identification. Therefore in the remainder of the paper we will refer to the repository system used in the use case as the "repository system".

The repository system described here is a university-wide digital asset management system with long-term archiving functions. It offers the possibility of archiving valuable assets, together with normalised metadata and content formats, offering multilingual access. Students, researchers and co-operators with the proper authorisation can upload and link the objects which, among others, can be text, image, and audio files in multiple formats. Searching and browsing of the contents is possible without logging in.

The repository system is used in many ways. It holds scans of precious books and incunabula, which can be accessed through a book viewer module. Projects run in the different institutes and faculties of the university archive their collections of audio recordings or historical documents in the repository system. The importer module allows creation of virtual collections of different content types that can be grouped, archived and published together.

Implementation The system consists of two main components, the backend and the frontend. The backend is realized with the use of Fedora Commons[1], which is an open source system that allows for storing, managing, and accessing digital objects. It also provides modules for searching (GSearch: Fedora Generic Search Service) and interfaces for the exchange of metadata (OAI-PMH - Open Access Initiative Protocol for Metadata Harvesting). The web frontend is responsible for the presentation of contents, or editing of metadata. The frontend was developed at the university. The communication between these two components is realised through the use of XML interfaces (REST-Calls).

Content Transformation The Fedora repository holds local content in the form of digital objects. The frontend interacts with the Fedora repository through the Fedora API, as seen in the Figure 4. The backend may also interact with other systems to obtain the content stored on different servers (distributed content) or may use web services to get additional information about the contents or to perform data transformation (e.g., format conversion, video streaming). These services are of particular interest, because they may change in different ways and therefore alter the information and the content delivered to the end users of the repository system.

In order to understand why services are so crucial for digital objects in a Fedora based repository, it is important

[1]http://www.fedora-commons.org

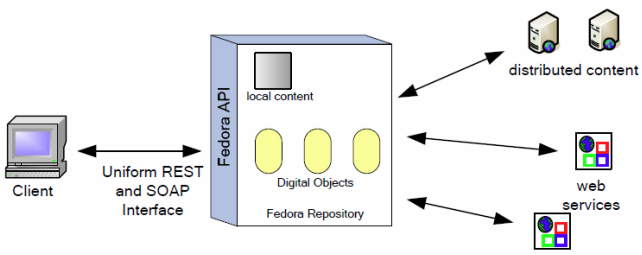

Figure 4: Backend (Fedora) as a mediator for services and content [6]

to understand the structure of a digital object. The digital object consists of four main parts: (1) a digital object identifier, (2) a descriptive part, including key metadata necessary to manage and discover the object and its relationships to other objects, (3) an item perspective which is the set of content or metadata items, and (4) a service perspective which provides methods for disseminating content.

Such a structure allows creating an object which has all data provided as static or dynamic data. For example, in the case of static data, a scientific paper can be stored in the Fedora repository in three formats: HTML, PDF, TEX and all of them will be grouped under one digital object. No disseminators (services performing operations on content) will be used to produce the content, because the content will be provided at the moment of creation of the digital object. However, the same final (visible to the end user) result could be obtained with the use of disseminators. It would be possible to store, for example, only the TEX file and use web services to generate on demand a PDF or HTML version of the document. This second solution allows saving storage space in the repository, but introduces dependency on the services (disseminators). Such dependencies are unavoidable in case of interactive contents like interactive art, games, computer programs, which need a special environment to render these artefacts.

Key functionalities We will discuss in the following section the key functionalities provided by the repository system. They have high impact on the content presented to the user and introduce other dependencies which may need to be considered during the risk analysis.

The Image Converter is used every time users access a web page with a summary about a digital document. For example, if the user browses through a collection of PDFs and opens one of them then they are presented with a preview of the first page of the paper which is a PNG file generated from the first page of the original file. This is achieved with the use of ImageMagick[2] and the corresponding Perl module which needs to be installed in the operating system underlying the repository system. If a different version of ImageMagick is used, it may happen that the conversion may result in a different output. Therefore, all of the dependencies of the repository system need to be documented carefully in order to be able to reproduce the same rendering.

The repository system also uses a streaming server which is

[2]http://www.imagemagick.org

run by the IT department of the university and is not a part of the repository system itself. When a video is accessed through the repository system there is a check if the video is already available at the streaming server. If it is, then the video is played to the user; otherwise the video has to be decoded by the server and then presented to the user. The video is displayed in a web browser window. In both cases, the streaming server has to be available. If it is not, or if it has changed (e.g. different codec library installed), the user may be presented with different rendering or, in the worst case, with no rendering at all.

3.2 Quality Assessment Workflow

Workflows have become popular as a means for specifying and automating computational experiments [5]. They serve a dual function: first, as detailed documentation of the executed process (i. e. the input sources and processing steps taken for the computation of a certain data item), and second, as re-usable, executable artefacts for data-intensive analysis. Using a workflow, a process is defined as a series of analysis steps which specify the flow of data between them.

We investigated a number of workflows published by third parties, many of them in the domain of digital preservation, such as workflows for file characterisation or format migration. The workflow that we will utilise as case study in this paper deals with duplicate detection in the book digitisation domain [8]. When scanning books, a software searches for errors in the scanned images. Due to the time lag of the error detection, errors are usually detected only when the scanning process has already scanned several more pages since the event occurred.

In such a case, the scanning process goes back to the erroneous page and restarts from there, thus re-scanning the erroneous and subsequently scanned pages. As the initial set of scans is not deleted, this leads to a set of duplicate scanned pages. The purpose of the duplicate detection is to perform a quality assurance of the document collection before the ingest into a repository system. The workflow specifically runs a duplication detection, and evaluates the performance of that duplication detection, with the help of a manually created ground truth that correctly identifies duplicate pages. Knowing the performance of the duplication detector is important to evaluate whether relying on the software solution only is sufficient, or a manual quality control step is needed in addition.

The workflow is authored in the Taverna workflow engine, and depicted in Figure 5. The actual duplicate detection is done via the *matchbox* application, developed as part of the SCAPE project. Matchbox (visible as the step "matchbox" in Figure 5) is implemented in Python, and called from the workflow via an external tool invocator, i.e. a system call. More specifically, the matchbox application is not available locally on the machine that executes the Taverna workflow, but is accessed as a remote service, via an SSH (secure shell) connection, on a different server. The output of the Matchbox algorithm is parsed in the "parse_matchbox_stdout" step, and the resulting matches as well as the log output of the application, are available as process outputs.

In order to evaluate the performance of the duplicate detection, reported matches are compared against a previously provided ground truth (passed via the process input "gt_filelist_path"), which contains the above mentioned true information on which pages are actual duplicates. This evaluation is implemented as a Java Beanshell script "matchbox_evaluate", which calls functions from a Java library provided in a JAR file to this processor step, and provides the workflow output "report", which combines correctly/incorrectly identified duplicates, missed duplicates, and measurements, such as precision, recall, and F-measure.

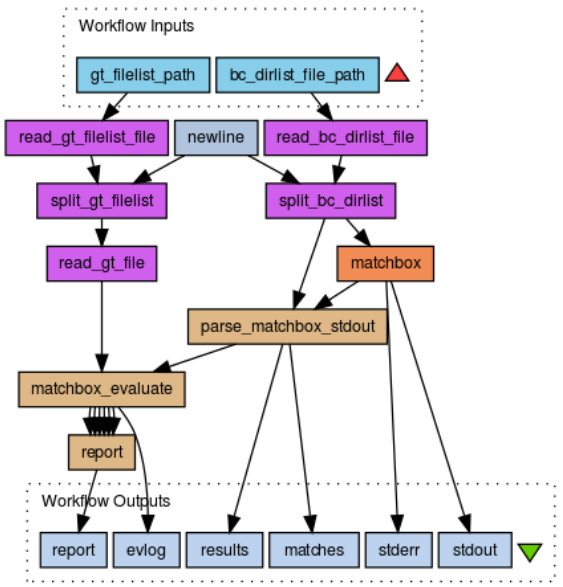

Figure 5: Duplicate Detection workflow, authored in the Taverna Workflow engine

While workflows are a step towards longevity and preservation of process executions, they themselves are not sufficient, as the execution environment, and external dependencies, are not properly addressed. In the use case example, an interesting aspect undermining that fact is the call to the Matchbox application, which is not performed via a local system call, but via an SSH connection on a remote server. Not only does that introduce a dependency on an external system, this call is also protected by the standard secure shell authentication mechanism, requiring a user name and password. To run the workflow, one thus additionally needs these credentials. Furthermore, the dependency on this external system means that the functioning of the workflow depends entirely on the functioning of that service.

Another interesting aspect is the step matchbox_evaluate, which, as mentioned above, is a Beanshell script that extensively uses an externally provided Java library for most of its functionality. This dependency to the Java library is declared in the workflow, but the actual library is not part of the workflow definition file, and thus has to be preserved separately. Another difficulty is the complex structure of the output of matchbox. All output information is returned in one text file, with a custom-defined format structuring the information on which pictures are identified as duplicates. Firstly, there is no documentation on the exact output format available, and in addition, the format was slightly changing throughout the different versions of development

of "matchbox".

The matchbox_evaluate component, which processes the output to do the evaluation, was developed by a different organisation than the duplicate detection itself, and the development of these two components was not always synchronised. Thus care has to be taken that the versions used of these two components are compatible to each other. Configuration management can help with this, but as there is not much information available on when the remote service would change its interface or structure and format of the returned result (such as at least notifications of a change), this issue is not easily resolved.

4. RISK ASSESSMENT

In this section, we explain how we followed the TIMBUS preservation framework described in Section 2.2 and depicted in Figure 3. We also present the results for both of the use cases.

4.1 Performing the assessment

Following the risk assesment process depicted in Figure 1 and described in Section 2.1, our first step was the identification of assets, events, risks and potential consequences. We utilized a combination of following techniques to collect this information:

- Checklist - list of risks previously defined, resulting from previous assessments with similar objectives. In this case, we used domain-specific lists, namely DRAMBORA and TRAC (Trustworthy Repositories Audit & Certification: Criteria and Checklist).
- System analysis - system/workflow documentation including its model and direct investigation of the running system/workflow. This technique involves analysing several processes performed by the system/workflow from different perspective (business view, infrastructure view)
- Brainstorming - it had the objective of identifying risks that were not detected from the checklists and system analysis.
- Semi-structured interviews - the risk assessment team met with the system operators to conduct individual interviews. They were asked a set of questions, encouraging them to look at a situation from a different perspective and thus identify new risks.
- Legal risks were analysed in accordance to the national and international legal documents by legal experts.

In the following steps, we analysed the risks and events. For each of the events we assigned the likelihood using a range of 5 values: very low, low, medium, high, very high. We used the same scale for assessing the consequences of the risks. Having done this, we created a risk matrix (see Figure 6). The dimensions of the matrix are likelihood and impact. By putting each risk into the cell that corresponds to its likelihood and impact, an overview of the severity of the risks was obtained.

The colours of the cells represent the risk level classes according to the established risk criteria, which represent the associated range of severity. This helped us to understand which risks need special attention when designing controls in the next step. The controls were designed for each of the risks identified and were applied to treat the risk, thus decreasing its severity. Naturally, the process of risk assessment needs to be periodically repeated in order to confirm that there are no new risks and if the controls are mitigating the risks efficiently.

Likelihood \ Impact	1	2	3	4	5
5			R03, R15		
4		R26, R14	R19, R23	R09	R32
3		R04, R06	R22, R25, R20	R01, R10, R17	R21, R07
2		R05, R22	R23, R24,R25	R33, R21	R24, R02, R12 ,R16, R18
1					

Figure 6: Risk Matrix for the repository case

4.2 Results

We applied the risk assessment process described in the above section to both of the use cases. In this section we present an overview of the results and explain the selection of controls.

Fedora Commons Based Repository We identified 29 events and 19 risks which constituted in total 46 pairs of events triggering the risks. We associated them with 5 different asset types, namely: organization, repository functionality, data stored in repository, repository system software. Such a wide variety of assets shows that the risk assessment focused not only on the risks related to digital preservation and software availability, but also on the risks related to the organizational context and legal issues. Table 1 shows some examples of pairs of events and risks identified, together with the assets they concern.

According to the risk assessment process we formulated controls for each of the risks and events. The results of this analysis were presented to the management of the repository and are going to be a base for the discussion on improvements to the repository system and its broad context. In the remainder of this section we would like to focus on a subset of controls which can be introduced using concepts and tools delivered by the TIMBUS project. Thus we demonstrate how the solutions from the digital preservation domain can mitigate a wide range of risks and not only those which are directly related to digital preservation.

Table 4.2 presents a list of TIMBUS related controls which can be used to minimize the likelihood of events or consequence of risks. It also depicts how their values change after control application. One can notice that some of the controls appear more than once in the controls column, e.g. *Context Model Instantiation*. Thus by applying one control we benefit from mitigating multiples risks. Furthermore, in the case of *Context Model Instantiation* we are controlling at one time events related to the typical digital preserva-

Table 1: Subset of assets, events and risks for the repository case

Asset	Event	Risk
Organization	Change of business model	Financial loss due to change of business model
Repository Functionality	Internal or external attacks	Functionality fault due to internal or external attack
Repository Functionality	Loss of expert knowledge	Functionality faults due to loss of expert knowledge
Organization	User's illicit activities	Reputation loss due to ilicit use of repository from user
Repository Software	Software faults	Software unavailability due to software faults
Data	Changes to content model	Loss of data integrity due to changes in data model

tion problems like *Environment changes*, but also business related risks like *Loss of expert knowledge* or *Changes in organizational structure*. In Section 5 we show how two of these controls were implemented, i.e. *Context Model Instantiation* and *External dependencies monitoring*.

Quality Assurance Workflow The subset of the risks identified for this use case is given in Table 4.2. It has to be noted that this use case has much fewer social and other contextual aspects to consider compared to the repository use case. This is due to the workflow itself being a mostly technical artefact, and the original environment where the workflow was executed being unknown and thus not a part of the use case. Thus, most risks concern technical aspects.

One important group of risks concerns the externally provided services, i.e. the duplicate detection algorithm. This service may be hosted outside of the organisation running the workflow, and outside of their control. Furthermore, the service is not following any protocol such as a WSDL web service, the communication, expected parameters and expected return values are not explicit in the workflow definition. Finally, the service requires an authentication, for which the user name and password are not kept along with the workflow definition.

Another group of risks is concerned with the library used in the workflow to process the results from the duplicate detection. This library can become unavailable, as it is not packed together with the workflow definition, or can have a fault, or can be incompatible with the version of the external service. The workflow engine itself is also a risk, as it may become unavailable, or incompatible with the operating system the workflow is deployed on. Finally, knowledge about the workflow setup, execution and interpretation is very often implicit, tacit knowledge of the owner of the workflow. If that person is not available anymore (e.g. due to changing jobs), it might not be possible to run the workflow anymore.

A subset of the identified controls for the workflow use case is given in Table 4.2. Some controls are the same or similar to the ones identified for the repository case; in general, similar observations as in the repository use case hold true: some of the controls appear more than once, thus mitigating more than one risk at the same time.

5. CONTROLS

In this section, we describe the selected controls and their application to the use cases in detail. The description explains how the control works, and in what way the risks or events are controlled. We selected the controls which address the entities on a high level of likelihood/consequence.

5.1 External dependencies monitoring

External dependencies monitoring is aimed at identifying the types of changes described in Section 2.3. Detecting this type of changes will not have an impact on the likelihood of the events happening, but it can help to reduce the consequence. This is on the one hand due to being able to detect a change earlier than when detecting it by a process triggered by the system itself. Potentially, we are able to detect and issue a fix to the issue before the problem surfaces in any process execution. Further, having monitored the service, we might have data available that allows us to more quickly identify the specific problem with the service, thus being able to find a solution for it quicker. External dependencies monitoring is applied in both use cases, as they both rely heavily on them.

Regarding the repository system, all of the services used are currently hosted within the infrastructure of the university. However, not all of them are under direct control of the repository system support team. Furthermore, due to the constant development of the repository system and provision of new services, it is likely that some of the services may be provided by external partners. The repository system can use any kind of web service regardless of its location. Such flexibility may cause potential threats. For example, when a service is down, many functions of the repository system depending on it will become unavailable or at least have their functionality limited. Furthermore, changes in the implementation of the external service may be unnoticed, but may impact the system. For this reason, we decided to monitor external services for their availability and changes and have response scenarios prepared in advance to mitigate the consequences of the service change.

We implemented the control using the Web Service Monitoring Framework (WSMF)[14]. The WSMF allows intercepting traffic communication between the system and the analysed web service. During standard operation of the service the data intercepted is stored as ground truth data. It is later used for validation of the service. We periodically sent the requests gathered in the ground truth data to the monitored web service and compared the responses with the ground truth data responses. On this basis we can detect whether the behaviour of the web services is changed. Figure 7 presents the application that allows performing these actions. For now we are able to detect changes in the Image Converter module of the repository system. The WSMF can also be applied to monitor web services responsible for conversion of content model by disseminators.

As shown in Table 4.2, the control *External dependencies monitoring* decreases the consequence of *Functionality faults*

Table 2: Subset of controls for the Repository use case

Control name	Control type	New Value	Old Value	Controlled Entity
List of users with administration rights	Likelihood	medium	high	Modification using administration rights
Context Model Instantiation	Consequence	medium	high	Changes in organizational structure
Context Model Instantiation	Consequence	medium	high	Loss of Expert Knowledge
External dependencies monitoring	Consequence	medium	high	Functionality faults
Group policies	Likelihood	medium	high	Modification using administration rights
Context Model (Infrastructure View)	Likelihood	low	medium	Environment changes
Mock-ups of services	Consequence	medium	high	Functionality fault
Preservation of system and data	Consequence	low	medium	Shortcomings in semantic understandability
Software escrow	Consequence	medium	high	Functionality fault
Substiution of missing components	Consequence	low	high	Functionality fault

Table 3: Risks identified for the Workflow use case

Event	Risk
Authentication failure	External service unavailability due to authentication failure
Correct Library version not found	Workflow execution failure due to library dependency unavailability
Data files not available	Workflow execution failure due to unavailability of data dependencies
Library faults	Workflow execution failure due to library dependency faults
Library unavailability	Workflow execution failure due to library dependency unavailability
Loss or lack of documentation	Shortcomings in semantic understandability due to loss or lack of documentation
External Service faults	Workflow execution failure due to external service dependency fault
External Service unavailability	Workflow execution failure due to external service dependency unavailability
Workflow engine faults	Workflow execution failure due to workflow engine fault
Workflow engine unavailable	Workflow execution failure due to workflow engine unavailability
Workflow executed on unsupported OS	Workflow execution failure due to unsupported operating system

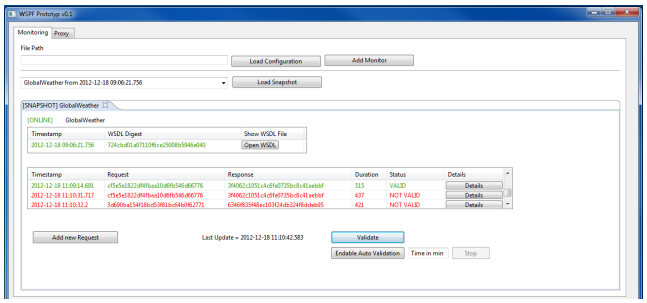

Figure 7: Web Service Monitoring Framework control panel [14]

from the high to the medium likelihood. This is because any potential changes influencing the functionality of the repository system are quickly noticed and instant preventive actions can be taken.

For the Workflow use case, the application is very similar - we can also apply the WSMF to the external service used, and are thus able to detect any changes at an earlier stage. Thus, we can reduce the consequences of external services becoming unavailable. According to Table 4.2 the consequence of *External service unavailability* is reduced from high to low.

5.2 Context Model Instantiation

The context model, introduced in [1] and described in Section 2.2 gives a comprehensive picture of the environment the process is embedded in. This allows for a documentation of the process steps, the actors, and their connection and dependency towards the technological infrastructure that provides the execution platform in a comprehensive and formal manner. Its formal representation enables reasoning, and checking for compliance.

The context model is an important control addressing a number of risks identified in our use cases. In some of these risks, having a context model that covers only the infrastructure aspects of our systems is enough, as these risks primarily deal with dependencies and conflicts between software applications. For some other risks, especially those regarding the knowledge on how the process is executed, a full-scale model that also covers application and business aspects is required.

Regarding the repository use case, even if the services are available locally the content presented to the user still may be altered in comparison to what was projected previously. Such a situation may occur when a user accesses some content (e.g., a video) which is rendered with different algorithms (e.g. different video codecs) and therefore may have a different look and feel of the digital object. In most cases, this is not a big issue for daily use, but in terms of digital preservation and documenting the significant properties of the digital object correctly for preservation purposes it is of great significance. Therefore when preserving a repository system, the knowledge about all of the elements impacting the final representation of the object have to be documented.

Also the dependencies of the repository system need careful documentation, because they also may affect the final result presented to the user. The repository system depends on many Perl modules and new implementations of modules may introduce changes in the behaviour of the system. Hence, it is crucial to maintain information about the software dependencies of the system in order to be able to recreate the same look and feel, as well as behaviour at any time in the future.

On-going development of the system, such as changes and enhancements of metadata schemas in order to enable Repository system to archive contents from various scientific disci-

Table 4: Controls for the Workflow use case

Control name	Control type	New Value	Old Value	Controlled Entity
Substitution of missing components	Consequence	low	high	Application dependency fault
External dependencies monitoring	Consequence	medium	high	Application license expired
Substitution of expired components	Consequence	low	high	Application license expired
Context Model (Infrastructure View)	Likelihood	low	medium	Application or Library incompatibility
Context Model (Infrastructure View)	Likelihood	low	medium	Application unavailability
Storing credentials for external services	Likelihood	medium	high	Authentication failure
Context Model (Infrastructure View)	Likelihood	low	medium	Correct Library version not found
External dependencies monitoring	Consequence	medium	high	Data files not available
Archiving and Preservation of data	Consequence	medium	high	Data files not available
Substitution of faulty components	Consequence	low	high	Library faults
Context Model (Infrastructure View)	Likelihood	low	medium	Library unavailability
Context Model Instantiation	Consequence	medium	high	Loss or lack of documentation
External dependencies monitoring	Consequence	medium	high	External Service faults
Mock-ups of services	Consequence	medium	high	External Service unavailability
Software escrow	Consequence	medium	high	External Service unavailability
Context Model (Infrastructure View)	Likelihood	low	medium	Workflow executed on unsupported OS

plines, creates another preservation requirement. For example, some digital objects may have been described through use of a metadata schema, which was later modified by adding new classifications and voluntary fields. However, it may happen that this new information cannot be added to the existing elements. These elements may then appear to a future user as corrupted, because the user may think that some of the metadata is missing despite the fact that the schema (the newer one) enforces its existence. The problem becomes even more complex when the concepts used in different versions of the schema are redefined and change their meanings. In order to prevent incorrect reasoning and wrong conclusions about the objects, it is essential to preserve the original versions of the metadata schemas and couple them with objects using them. All of these can be described in the Context Model. Due to the non disclosure agreements we are not allowed to present the example of the Context Model for the repository case.

Concerning the open source workflow use case, the technical part of the context model is an effective control regarding dependency and incompatibility risks. With the concepts provided by the meta-model, we can formally capture the dependencies between the application and library components used in the system. This helps when identifying issues that could be caused by changing versions of certain parts of the system setup. Furthermore, by having the full instantiation of the context model, it becomes clear what sequence of steps is needed to be carried out in the process, and how each step is supported by certain parts of the infrastructure. Also, existence of external services become clear, and their impact to certain parts of the process is explicit. The data flow between the steps is formally defined, which helps in understanding how the data is processed.

A simplified version of a corresponding instance of the TIMBUS Context Model is depicted in Figure 8. The model depicts the external system that is called via SSH. It also shows the third-party library that is needed for the matchbox algorithm evaluation.

5.3 Application Substitution

An application is usually utilised to manipulate or render a digital object. By replacing (substituting) the original application interpreting the digital object the functionality

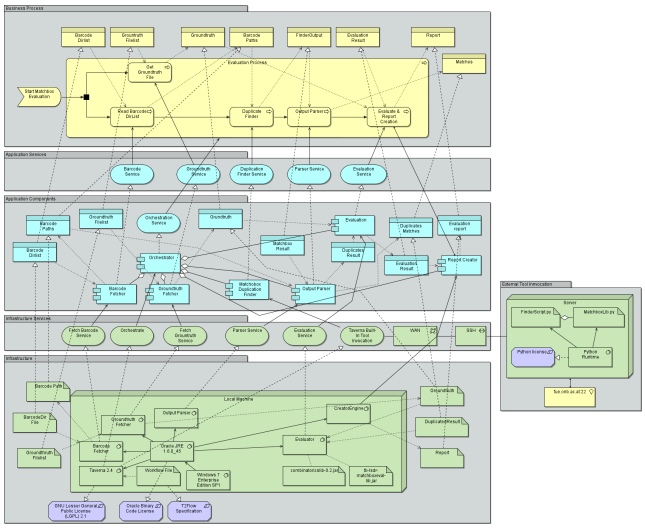

Figure 8: Context Model of the Duplicate Detection workflow

of this application is emulated. This is an effective control to mitigate risks that can stem from faulty or incompatible software applications, libraries and components utilised in the system. By replacing them with another component that provides equivalent behaviour, but does not exhibit the risks, we successfully mitigated that risk by application of emulation.

As part of the TIMBUS project, we developed a service that allows for automatic identification of potential alternative software implementations, and thus application emulation. The service is built around knowledge bases obtained from linked data sources such as Freebase[3][4], as well as software packages as they are present often in Linux operating systems, where *virtual packages* provide a categorisation of packages that provide the same functionality. The service then operates on a representation of the system, authored by using the context model, and proposes potential replacements, that in turn should be analysed by a digital preservation expert for their usefulness and feasibility.

[3]http://www.freebase.com/

This approach can be used in the case of the repository system to decrease the consequence of *Functionality fault* risk (see Table 4.2). For example if the Tomcat application server that is a container for most of the repository backend is obsolete and loses the community support, it can be replaced with a compatible one, like Jetty, which may not have this problem. Moreover, multiple Java and Perl libraries may also need to be substituted. One of the potential reasons could be low security of the component, then such a vulnerable library may be replaced with a recommended alternative. This shows again that the digital preservation tools can also ease day to day maintenance of the system.

6. CONCLUSIONS

This paper describes the results of a case study conducted on two use cases: a repository system and an automated workflow representing typical digital preservation quality assessment processes. We followed the TIMBUS preservation framework and risk assessment process defined by ISO 31000:2009 to identify potential risks and their impact on the sustainability of systems and workflows.

The case study revealed a wide range of risks affecting not only the technical aspects of the cases but also organizational aspects. First and foremost, it confirmed the concerns that the repository systems may need to undergo several digital preservation actions. Hence, there should be more attention to this problem within the digital preservation community and the contents of the repositories are not the only thing we need to worry about. Furthermore, the preservability of both systems and workflows is endangered due to a high dependence on external services and insufficient documentation of their dependencies.

Using tools developed within the TIMBUS project we demonstrated how these risks can be substantially mitigated. We used the external dependencies monitoring, context model instantiation and application emulation as controls to achieve this aim.

ACKNOWLEDGMENTS

This research was co-funded by COMET K1, FFG - Austrian Research Promotion Agency, by the FCT - Fundação para a Ciência e a Tecnologia, under project PEstOE/EEI/LA0021/2013, and by the European Commission under the IST Programme of the 7th FP for RTD - Project ICT 269940/TIMBUS.

7. REFERENCES

[1] G. Antunes, M. Bakhshandeh, R. Mayer, J. Borbinha, and A. Caetano. Using ontologies for enterprise architecture analysis. In *Proceedings of the 8th Trends in Enterprise Architecture Research Workshop (TEAR), in conjunction with the 17th IEEE International EDOC Conference*, Vancouver, Canada, September 9-13 2013.

[2] J. Barateiro. *A Risk Management Framework Applied to Digital Preservation*. PhD thesis, Universidade Técnica de Lisboa, Instituto Superior Técnico, 2012.

[3] J. Barateiro, G. Antunes, F. Freitas, and J. Borbinha. Designing digital preservation solutions: a risk management based approach. *The International Journal of Digital Curation*, 5:4–17, 2010.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[5] E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, May 2009.

[6] Fedora Commons Community. Fedora commons tutorial 2: Getting started: Creating fedora objects using the content model architecture. Technical report, 2007.

[7] T. O. Group. *ArchiMate 2.0 Specification*. Van Haren Publishing, 2012.

[8] R. Huber-Moerk, A. Schindler, and S. Schlarb. Duplicate detection for quality assurance of document image collections. In *Proceedings of the 9th International Conference on Digital Preservation (IPres2012)*, Toronto, Canada, October 1-5 2012.

[9] ISO. *ISO Guide 73:2009 – Risk management – Vocabulary*. International Organization for Standardization, 2009.

[10] ISO. *ISO 16363:2012 – Space data and information transfer systems – Audit and certification of trustworthy digital repositories*. International Organization for Standardization, 2012.

[11] ISO/FDIS. *ISO/FDIS 31000:2009 – Risk management – Principles and guidelines*. International Organization for Standardization, 2009.

[12] Y. Li and M. Banach. Institutional repositories and digital preservation: Assessing current practices at research libraries. *D-Lib Magazine*, 17(5/6), 2011.

[13] A. McHugh, R. Ruusalepp, S. Ross, and H. Hofman. The digital repository audit method based on risk assessment (DRAMBORA). In Digital Curation Center and Digital Presevation Europe, 2007.

[14] T. Miksa, R. Mayer, and A. Rauber. Ensuring sustainability of web services dependent processes. *International Journal of Computational Science and Engineering (IJCSE)*, 2014. Accepted for publication.

[15] S. Strodl, D. Draws, G. Antunes, and A. Rauber. Business process preservation: How to capture, document & evaluate? In *Proceedings of the 9th International Conference on Preservation of Digital Objects*, Toronto, Canada, 1–5 October 2012.

[16] S. Strodl, R. Mayer, G. Antunes, D. Draws, and A. Rauber. Digital preservation of a process and its application to e-science experiments. In *Proceedings of the 10th International Conference on Preservation of Digital Objects*, Lisbon, Portugal, September 2013.

[17] M. Van der Graaf and L. Waaijers. *A Surfboard for Riding the Wave. Towards a four country action programme on research data. A Knowledge Exchange Report.* 2011.