SERIES	TIMBUS WHITE PAPERS
Title	Verification and Validation of Preserved and Redeployed Business Processes

TIMBUS WHITE PAPER:

Verification and Validation of Preserved and Redeployed Business Processes

Publication Date: 20.09.2014

Dissemination Level: PU

SERIES	TIMBUS WHITE PAPERS
Title	Verification and Validation of Preserved and Redeployed Business Processes

TIMBUS is supported by the European Union
under the 7th Framework Programme
for research and technological development and demonstration activities (FP7/2007-2013)
under grant agreement no. 269940

SERIES	TIMBUS WHITE PAPERS
Title	Verification and Validation of Preserved and Redeployed Business Processes

Authors		
Name	**Organisation**	**e-mail**
Tomasz Miksa	SBA	tmiksa@sba-research.org

Contributors		
Name	**Organisation**	**e-mail**

Internal Reviewer		
Name	**Organisation**	**e-mail**
Perumal Kuppuudaiyar	Intel	perumal.kuppuudaiyar@intel.com

Disclaimer

Table of Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
BP	Business process
DP	Digital Preservation
DIO	Domain Independent Ontology
DSO	Domain Specific Ontology
OWL	Web Ontology Language
QA	Quality Assurance
RDF	Resource Description Framework
SPARQL	Simple Protocol and RDF (Resource Description Framework) Query Language
UML	Unified Modelling Language
VM	Virtual machine

1 Executive Summary

Preserving processes requires not only to identify all process components, but also to intercept all interactions of the process with external influencers. In order to verify if the collected data is sufficient for the purpose of redeployment, as well as to verify that the redeployed process performs according to expectations, a framework for verification is needed. For this reason we devised the VFramework which provides the guidelines for successful verification of preserved and redeployed processes. The information collected during the VFramework application is stored into the VPlan, which is an ontology for a systematic organizing and storing of verification information. We use a set of SPARQL queries not only to validate the VPlan by checking if the required information is complete, but also to create reports presenting the verification information in a coherent way, thus easing the analysis of the results of the verification and validation process. Finally, we describe tools that were implemented in order to automate comparison of metrics used during the assessment. The tools enable PNG, PDF and TEX comparison. The requirements for the tools were driven by the use cases, on which the VFramework was successfully applied. This white paper introduces the concepts used by us during the verification. For more details please refer to the (TIMBUS Consortium, 2013) and our proceedings papers (Miksa et. al, 2013) and (Miksa et. al, 2014).

2 Introduction

Traditionally, research in the area of digital preservation deals with preservation of static information like documents, scans, and other kinds of data. The long term preservation of entire systems and processes was not in the centre of attention. Addressing this new challenge requires advanced methods and processes which ensure that the process context is described adequately. This includes the collection of sufficient information of all involved components, which enables future redeployment.

No matter how well-engineered the process for preservation of processes is, it cannot guarantee that all necessary information required to run the process was recorded. Given the complexity of preserving entire systems and processes, we thus need to derive means for reliably verifying whether a process being re-deployed performs correctly according to preservation goals. We need to ensure that not only sufficient information is collected during planning and preserving of the process, but also to confirm that the redeployed process performs according to the expectations of the redeployment scenario.

The verification of redeployed processes is a complex task which may vary in its form due to several factors:

- the way the processes are specified,
- the drivers for their preservation,
- the preservation strategies applied,
- the reasons for the redeployment,
- the redeployment environments, etc.

However, regardless of these differences, all processes must be verified for measuring the success of the redeployment. Otherwise, there is no guarantee that the process running in the redeployed environment is the one which was meant to be redeployed. Such evidence is crucial in litigation cases when the correctness of the original process, executed at some time in the past, could be questioned, and the only way to check this is to re-run the original process. In such cases, the method for verification of redeployed processes should provide irrefutable evidence that the redeployed process is behaving exactly the same way as the original.

In this white paper we present the VFramework which defines the framework for verification of preserved and redeployed processes. We also describe the VPlan which is an ontology used to store the information collected during the VFramework application. Then we describe in what way SPARQL queries facilitate validation of data stored in the VPlan and also ease the analysis of data by provision of well-structured reports. Finally, we provide an overview of tools developed for automatic comparison of metrics that are used during the verification.

3 VFramework

The VFramework was created to verify that a redeployed process performs according to expectations. The framework's foundation is driven by two major requirements.

3.1 Requirements

Firstly, the framework has to be independent of the situation in which different digital preservation actions were applied to the full process or to different parts of the process. In such situations some of the process' parts may be substituted, re-engineered, emulated, migrated, etc. As a result, the redeployed process which is to be verified is not necessarily an exact copy of the original process.

The framework has to be capable of verifying the execution of similar processes or their parts. By similarity of processes we mean a situation, in which the functionality or characteristics of the process have been altered, but the deviation is either desired (e.g. faster computation) or acceptable (e.g. some functionality is limited but for the purpose of redeployment it is not required). Such situations may be an inevitable side effect of the digital preservation actions or a consequence of deliberate actions (e.g. improved implementation of the process). The framework has to support such situations regardless of its origin, and be capable of evaluating full and partial redeployments of processes.

Secondly, due to the high variety of the nature and implementation of the processes and a wide range of potential user requirements that had to be considered, the framework has to be flexible to cover all these requirements and settings. Therefore it has to remain at a relatively high level of abstraction and be customizable for the concrete processes which are going to be preserved. The guidance on customization is provided by the framework in order to achieve the comprehensiveness of the process verification.

3.2 Solution overview and definitions

The VFramework is depicted in Figure 1 and consists of two sequences of actions. The first one (depicted in blue) is performed in the original environment. The result obtained from execution of each step is stored in the VPlan (see Section 4). The second sequence (depicted in green) is performed in the redeployment environment. The necessary information for each of the steps is obtained from the VPlan.

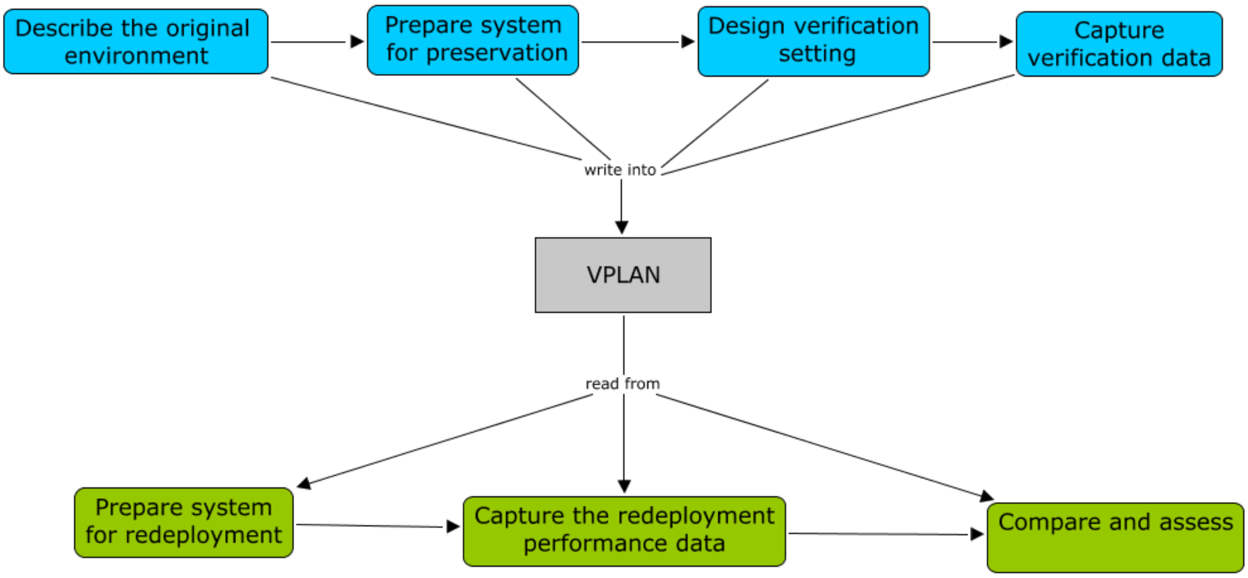

Figure 1: VFramework.

Original environment denotes the system in which the process that is going to be preserved is deployed and operates. The redeployment environment is the system in which the process will be installed once the decision to redeploy the preserved process is made. It is very likely, that the redeployment will take place at some distant time in the future, when the original environment does not exist anymore and the process may need to be re-engineered to fit it into a new system.

Apart from descriptive metadata, the VFramework uses two kinds of data: verification data and redeployment performance data. The verification data is collected during the execution of the process in the original environment. It provides information on details of the execution of process instances, focusing on measuring significant properties. Interactions with external components have to be stored as well. For this purpose, external interaction data being part of verification data is collected. This external interaction data represents a record of all interactions of the process with external components during the execution of a specific process instance in a scenario to be used for verification. This data is reapplied in the redeployment environment to ensure determinism, by recreating the same external interactions. The redeployment performance data is collected during the execution of the process in the redeployment environment. It provides information on details of the execution of the process instances, focusing on measuring significant properties. It is used for comparison with verification data to assess the redeployment. The steps of the framework are described below and in (Miksa et al., 2013).

3.3 VFramework steps detailed

Here we explain the steps of the VFramework. For more details please refer to (TIMBUS Consortium, 2013) or (Miksa et. al, 2013).

VFramework step 1: Describe the original environment

The aim of this step is to describe the process and document its context by identifying environment dependencies in which the process is deployed. Information on:

- the motivation for the preservation of the process considered,
- the redeployment scenarios,
- set of example instances to be used for verification,

is collected. This corresponds largely to the steps 1-3 of the "Define Requirements" phase in preservation planning (Becker et al., 2008), with the first step being subdivided into two more fine-grained steps. The process is described using the TIMBUS Context Model. An example is presented in Figure 2.

VFramework step 2: Prepare system for preservation

The aim of this step is to identify the interactions of the process, i.e. all inputs and outputs of the process, but also configurations of process parameters, as well as influences of other components sharing the process environment or used indirectly by process components. This information is needed in order to ensure deterministic execution of the process and thus ensure reliable assessment. The steps should be conducted in view of redeployment scenarios and significant properties defined for the process in step 1.

VFramework step 3: Design verification setting

The aim of this step is to identify the measurement points of the process, specify metrics used to assess quality of preservation actions and couple them with thresholds which are used as criteria for the assessment. The measurement points can be defined as points of the process where data enabling reasoning about correctness of the process execution is collected. The investigation should be conducted in the view of redeployment scenarios and significant properties defined for the process.

VFramework step 4: Capture verification data

This step has two main tasks:

- configuration of the capturing environment for collection of verification data,

- collection of the verification data while the process is monitored by tools which trace process interactions.

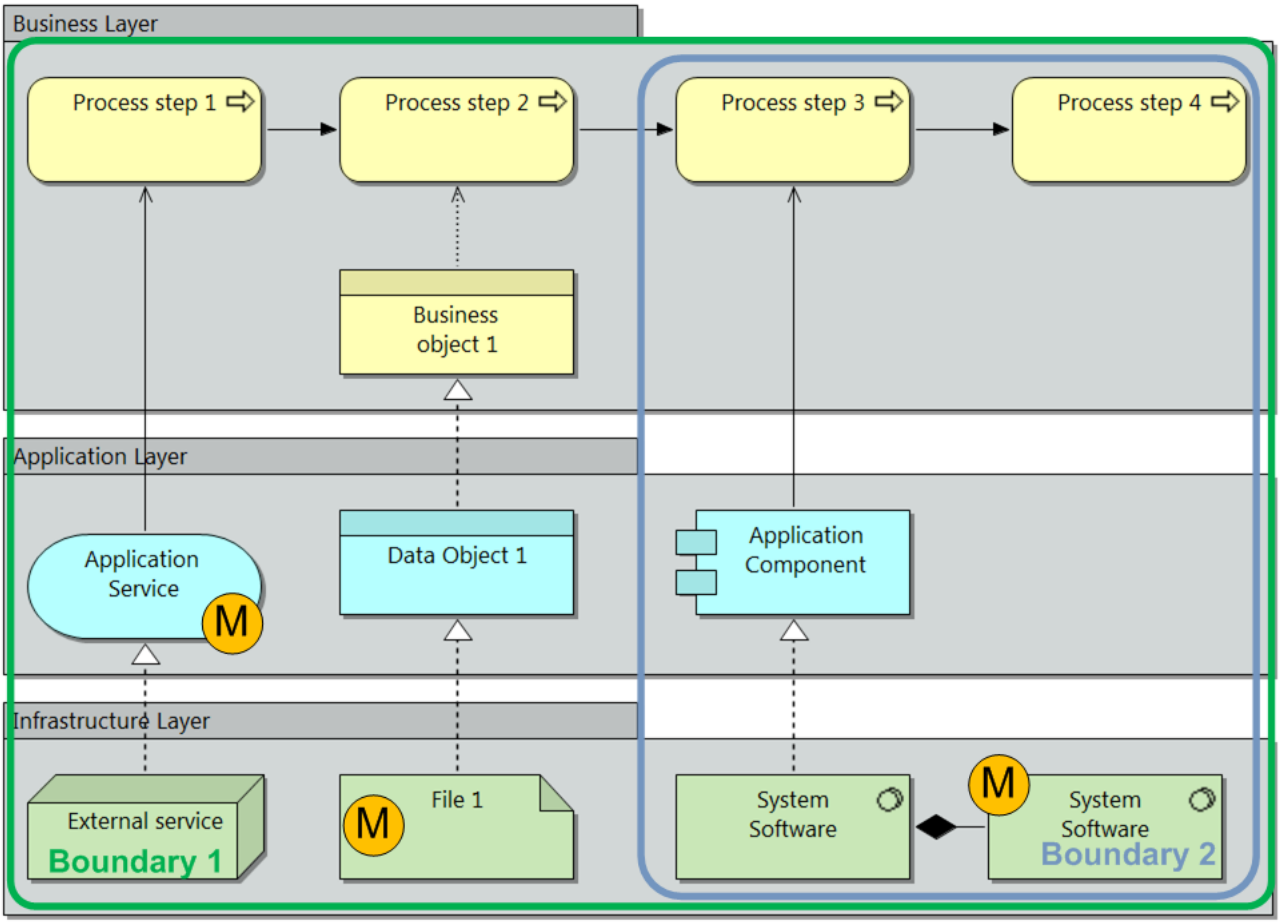

Figure 2: Example of a process modelled in ArchiMate depicting basic concepts of the VFramework. There are two boundaries (green and blue line marking elements belonging to each of them) and three measurement points (orange circles with letter "M" inside).

VFramework step 5: Prepare system for redeployment

This is the first step performed in the redeployment environment. This step has three main objectives:

- configuration of the redeployment environment for collection of redeployment performance data,
- redeployment of the process in a new environment,
- execution of the process instances.

VFramework step 6: Capture redeployment performance data

The aim of this step is to collect the redeployment performance data from the new system and verify if the data collection conditions were fulfilled.

VFramework step 7: Compare and assess

The comparison of significant properties measured in both environments is conducted in this step. The comparison is described in a report and a decision about fulfilment of redeployment purposes is made.

4 VPlan

The VPlan is created when the original process is preserved and accessed during the redeployment phase. It was designed to handle the information collected by the VFramework application. A VPlan is created per process and it contains process instances which can verify particular process execution. The VPlan is publically available at:

https://timbus.teco.edu/svn/public/ontologies/VPlan.owl.

Figure 3 depicts the concept map of the VPlan. The light blue boxes are the classes, e.g. *VPlan*, *Metric*, *RedeploymentScenario*, etc. The named arrows connecting the light blue boxes are object properties relating classes to each other, e.g. *measures*, *appliesToScenario*, *hasInstance*, etc. The arrows which point to the green boxes are data properties, these are namely: *isLocatedAt*, *hasTextDescription* and *isInline*. There are also five dark blue boxes, which are individuals used for creating an enumeration for the *MetricTargetOperator* class. Finally, there are 3 grey boxes which depict elements imported to the VPlan by importing the Context Model.

In general the VPlan links the requirements expressed by significant properties and metrics with the way they are measured. To describe the measurement process the information on process instances and capturing processes is provided. The VPlan uses the Context Model to precisely depict from which process' part the information was captured. Moreover, it includes also the capturing processes, which were originally modelled in Archi and later converted to an OWL ontology in order to document the way the data was collected. Finally, the VPlan stores not only information on data location used to run the process, but also the data which was captured from the process. These relations were schematically presented in Figure 4.

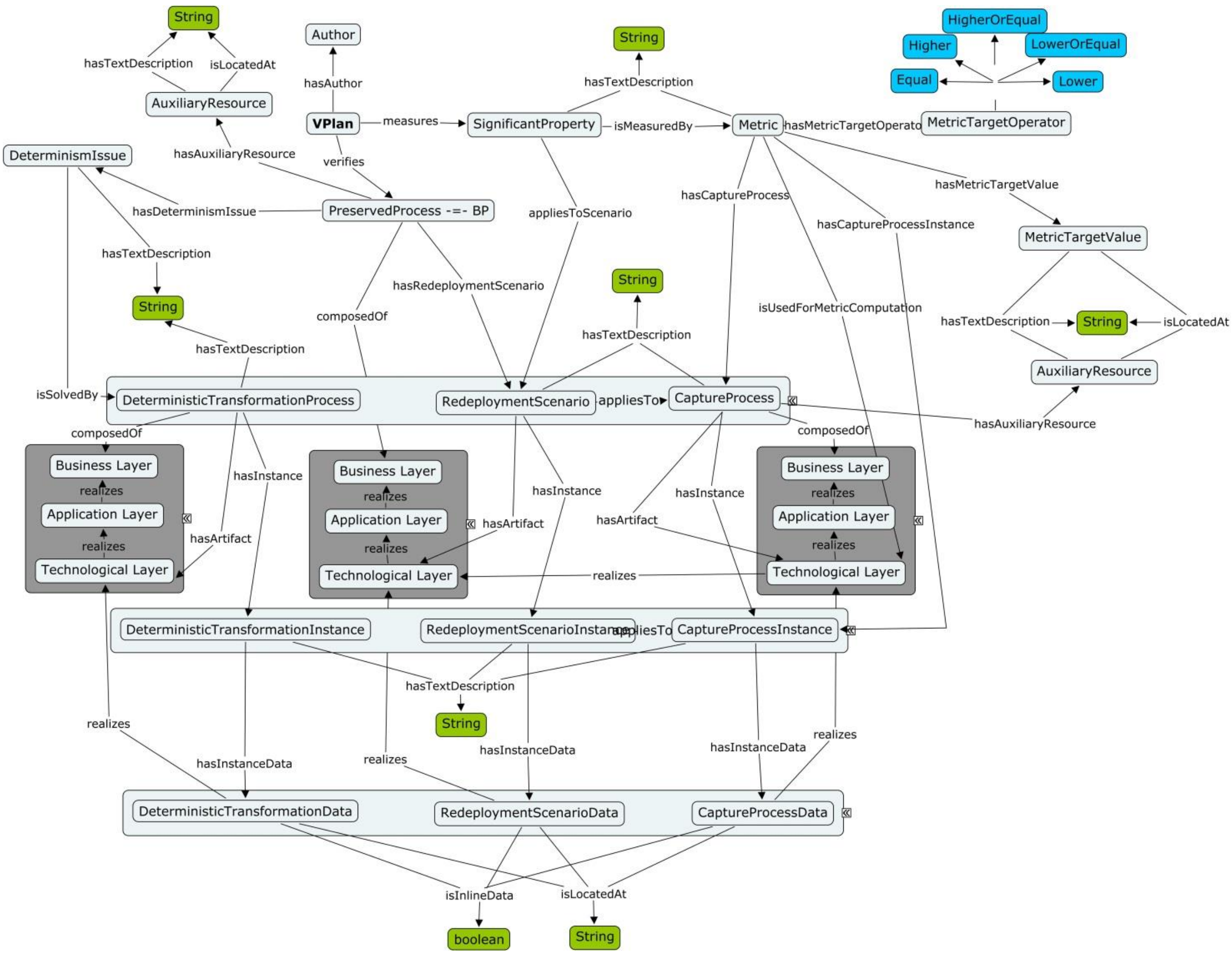

Figure 3: VPlan concept map depicting class, object and data properties.

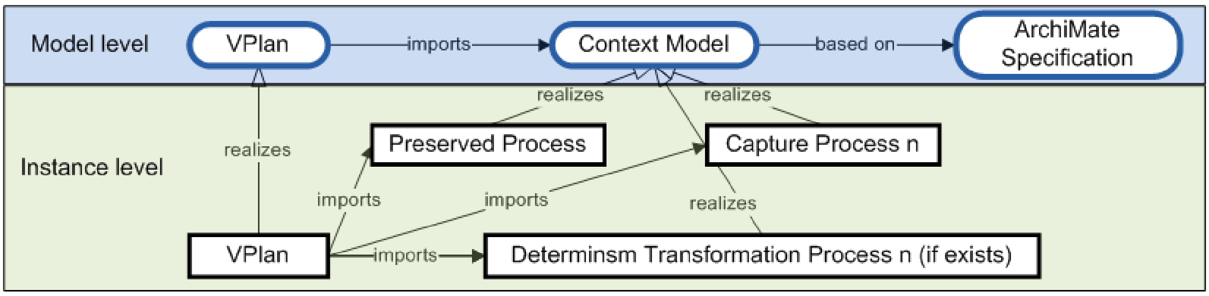

Figure 4: Relation of the VPlan to the Context Model

5 SPARQL queries

SPARQL[1] is a query language that allows retrieving RDF[2] data. The VPlan and the TIMBUS context model are RDF compatible and for this reason the SPARQL queries can be used in the process of verification and validation. This section presents sample queries and highlights the benefits they bring on various stages of the VFramework application.

5.1 Validation of the VPlan

When the data is collected in the original environment (step 4 of the VFramework) it is essential to validate it. This requires manual or automatic actions and varies depending on the data type. In the proposed solution the VPlan plays a central role of a deposit in which all the information about the preserved business process is stored. Furthermore, in case when only parts of the VPlan are automatically generated and the rest is created manually there is a risk of errors in the model introduced by a human. Therefore, the validation of the VPlan is also needed.

The SPARQL queries can be used for validation of the VPlan. The combination of queries is capable of checking the model completeness, i.e. detecting if the object or data properties of the VPlan instance comply with the VPlan specification, or if the object properties between the individuals of the VPlan instance are allowed. In Figure 5 an example of validation query is presented.

The query validates the instance of the VPlan. In this particular case the individual of a class Metric which name contains “SP1M1” is validated, i.e. it is checked if this individual has all its properties specified according to the VPlan specification of the class Metric. The example execution of the query is presented in the Figure 5. One can notice, that the result revealed that two properties are missing, namely: *VPlan:isUsedForMetricComputation* and *VPlan:hasMetricTargetOperator*. Hence, the VPlan is not valid and this issue needs to be fixed.

[1] SPARQL: http://www.w3.org/TR/rdf-sparql-query/

[2] RDF: http://www.w3.org/RDF/

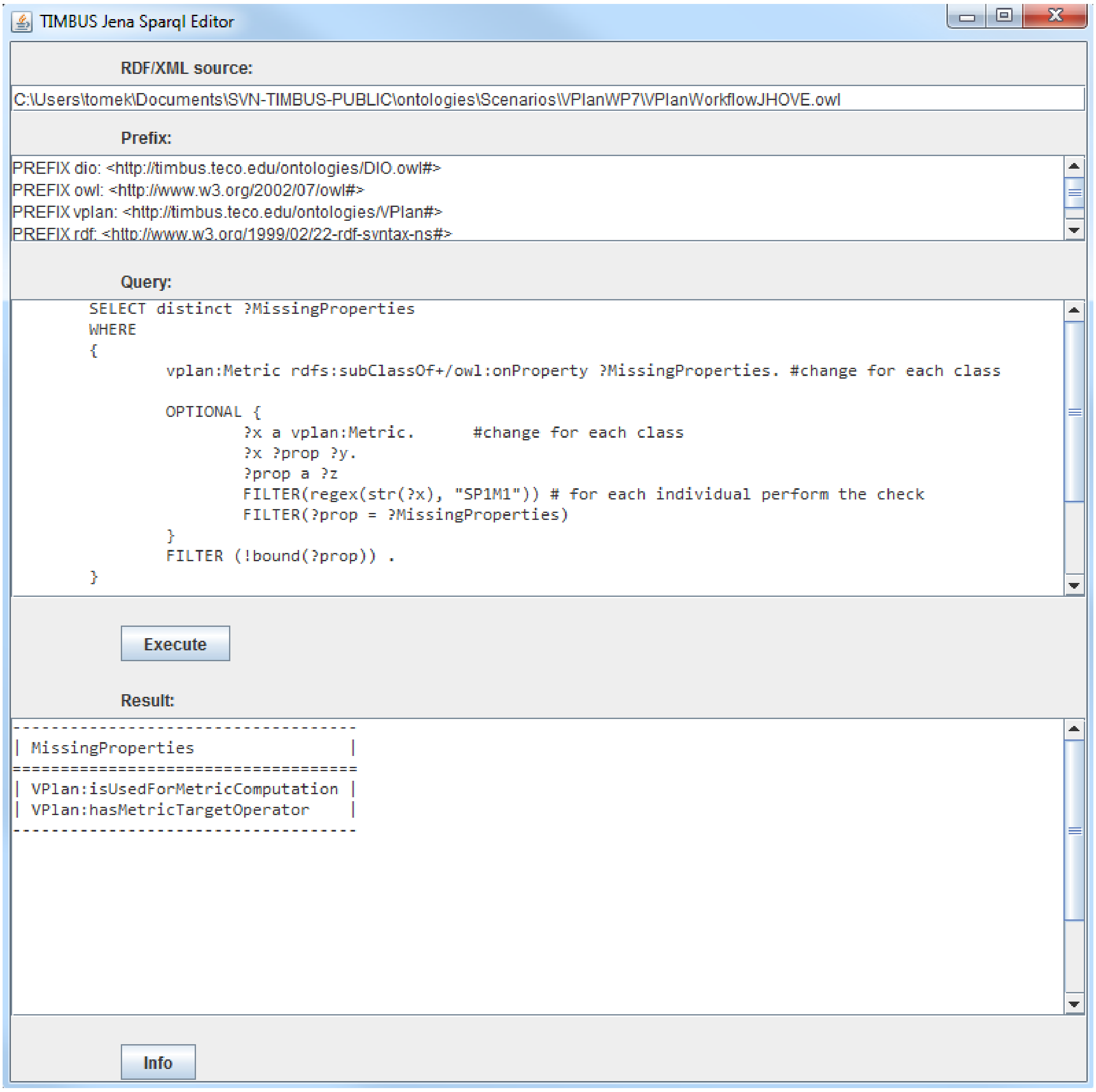

Figure 5: Example of SPARQL validation query execution. The result reveals that at least two properties are not defined in the VPlan instance.

5.2 Reporting

SPARQL queries can also facilitate presentation of data stored in the VPlan. During the redeployment phase of the VFramework the preservation expert must make many decisions concerning the redeployment using the information about the original process. For example, the capturing processes in the redeployment environment have to mimic the capturing processes from the original environment. Furthermore, when in the last step of the VFramework the assessment of the redeployment is performed, the information on: metrics, target values and expected values, in the redeployment are needed. In both cases this information was defined during the first phase of the VFramework and is stored within the VPlan. Hence, at the different stages of the VFramework application needs to obtain different information from the VPlan. Using of SPARQL queries is a convenient way of doing this. The SPARQL queries executed on the VPlan can provide answers to the following questions (and are not limited to them only).

- What are the significant properties of the process?
- What are the metrics used to measure the significant properties?
- What is the capture process for each of the metrics?
- Where are the auxiliary resources fostering understanding of the capture process located?
- Where are the measurement points in the process?
- Which data has been collected for calculating metrics?
- What are the expected values of the metrics?
- Where is the captured data?
- What instances of the process are used for verification?
- Are there any determinism issues in the process?
- What are the process steps?
- Are there any auxiliary descriptions of the process?

Examples of queries can be found in the Annex A of (TIMBUS Consortium, 2013).

6 VFrame Comparators

During the evaluation of the VFramework we applied it to various use cases. One of the use cases focused on a scientific process within a civil engineering domain that required data processing, transformation and visualisation. According to the VFramework we have defined a set of significant properties and metrics that are measured in order to evaluate whether the significant property is fulfilled. Hence, we developed a set of tools that enabled us to automate this process. We call this tools suite VFrame Comparators. They are based on publically available open-source libraries and are implemented in Java. The VFrame comparators consist of 3 tools which are briefly characterized below and described in details in the Annex B of (TIMBUS Consortium, 2013).

6.1 *PDF Document Comparator*

The comparator takes two different pdf documents as its input. As a result:

- it constructs a third pdf file with differences highlighted using diff-pdf[3],
- extracts pdf documents metadata and fixities and produce comparison table,
- extracts texts from pdf files and compare texts and then output them in html file format by highlighting differences in texts.

It provides outputs in form a visual list of differences between the contents of the document (see Figure 6). Furthermore, it provides also a table depicting changes in the metadata of the DPF document as presented in Table 2.

See the output of `diff-pdf --help` for complete list of options.

Some text	Head1	fgsdg	gsgsgs
s	erdff	34636	45764
dfsdf	fasdf	6363	47474524 5

Figure 6: Example of PDF visual comparison

[3] Diff-pdf tools for visual pdf comparison: https://github.com/vslavik/diff-pdf

Table 1: PDF Comparison Sample Output.

VVFrame PDF Metadata Comparison

PDF Metadata	PDF-1	PDF-2	Difference
Word count	476	476	No
dcterms:modified	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
meta:creation-date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
meta:save-date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
dc:creator	Kuppuudaiyar, Perumal	Kuppuudaiyar, Perumal	No
Last-Modified	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
Author	Kuppuudaiyar, Perumal	Kuppuudaiyar, Perumal	No
dcterms:created	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
Complex words	81	81	No
date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
modified	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
creator	Kuppuudaiyar, Perumal	Kuppuudaiyar, Perumal	No
xmpTPg:NPages	4	4	No
Creation-Date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes
meta:author	Kuppuudaiyar, Perumal	Kuppuudaiyar, Perumal	No
created	Thu Sep 19 11:18:18 BST 2013	Thu Sep 19 11:19:35 BST 2013	Yes
producer	Microsoft® Word 2010	Microsoft® Word 2010	No
CheckSum	4dca60f0ef1fed81150560f5edfb44107cd021e0	e4df74b2802e1979a4ba681ea130ce9ecce2e0b8	Yes
Sentence count	51	51	No
xmp:CreatorTool	Microsoft® Word 2010	Microsoft® Word 2010	No
Content-Type	application/pdf	application/pdf	No
Last-Save-Date	2013-09-19T10:18:18Z	2013-09-19T10:19:35Z	Yes

6.2 Latex Document Comparator

This comparator takes two input LaTeX documents and analyses them. As a result an XML file describing differences is produced. The output of the tool is depicted in Figure 7. It shows positions of words which were deleted and inserted.


```
  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <latexDifference>
  - <deletes count="5">
      <delete index="3598">deleted here</delete>
      <delete index="3279">gesfsefsereat</delete>
      <delete index="567">Example</delete>
      <delete index="8269">which should adding wefjef efkejfqkefj efqejfqef efqefqfqf</delete>
      <delete index="2823">some dummpy inserted</delete>
    </deletes>
  - <inserts count="7">
      <insert index="3734">(blaaaa)</insert>
      <insert index="3143">difference</insert>
      <insert index="3463">comments</insert>
      <insert index="608">Example111</insert>
      <insert index="3645">of things</insert>
      <insert index="3326">great</insert>
      <insert index="2648">by</insert>
    </inserts>
  </latexDifference>
```


Figure 7: Latex Document Compare sample output.

6.3 Image (PNG) Comparator

The comparator takes two different image files of the same size for its input. Then it extracts key features from those two images and as result produces:

- third image file with differences only pixels,
- summary table of differences comparing extracted key features.

A sample output of the comparator is presented in Table 2.

Table 2: Sample Image Comparison Output Table.

VVFrame Image Features Comparison

Image Features	Image-1	Image-2	Difference
Image	test-images\test_i...	test-images\test_i...	Yes
Format	JPEG (Joint Photo...	JPEG (Joint Photog...	No
Class	DirectClass	DirectClass	No
Geometry	640x480+0+0	640x480+0+0	No
Resolution	72x72	72x72	No
Print size	8.88889x6.66667	8.88889x6.66667	No
Units	PixelsPerInch	PixelsPerInch	No
Type	TrueColor	TrueColor	No
Endianess	Undefined	Undefined	No
Colorspace	sRGB	sRGB	No
Depth	8-bit	8-bit	No
Channel depth:			
red	8-bit	8-bit	No
green	8-bit	8-bit	No
blue	8-bit	8-bit	No
Channel statistics:			
Red:			
min	0 (0)	0 (0)	No
max	255 (1)	255 (1)	No
mean	39.0861 (0.153279)	36.6061 (0.143553)	Yes
standard deviati...	85.0002 (0.333334)	82.8166 (0.324771)	Yes
kurtosis	1.86437	2.39645	Yes
skewness	1.92713	2.05607	Yes
Green:			
min	0 (0)	0 (0)	No
max	255 (1)	255 (1)	No
mean	28.3792 (0.111291)	30.0772 (0.11795)	Yes
standard deviati...	69.2535 (0.271582)	71.8709 (0.281847)	Yes
kurtosis	4.7122	3.91256	Yes
skewness	2.50339	2.3536	Yes
Blue:			
min	0 (0)	0 (0)	No
max	255 (1)	255 (1)	No
mean	19.6895 (0.077213...	20.037 (0.0785766)	Yes
standard deviati...	57.2916 (0.224673)	58.3125 (0.228676)	Yes
kurtosis	10.7709	10.1926	Yes
skewness	3.46997	3.39328	Yes
Image statistics:			
Overall:			

7 Bibliography

Miksa, T., Vieira, R., Barateiro, J., & Rauber, A. (2014). VPlan - ontology for collection of process verification data. *Proceedings of the 11th International Conference on Digital Preservation (iPres 2014).* Melbourne.

Miksa, T., Vieira, R., Rauber, A., Proell, S., Strodl, S., & Barateiro, J. (2013, 9). Framework for Verification of Preserved and Redeployed Processes. *Proceedings of the 10th International Conference on Digital Preservation (iPres 2013).* Lisbon.

TIMBUS Consortium. (2013). *D4.7: Validation of DP'ed Business Processes & Verification of Redeployed Business Processes.*